Secure Location Sharing

Mahdi Zamani
Department of Computer Science
University of New Mexico
Albuquerque, NM, USA
zamani@cs.unm.edu

Mahnush Movahedi
Department of Computer Science
University of New Mexico
Albuquerque, NM, USA
movahedi@cs.unm.edu

ABSTRACT

In the last decade, the number of location-aware mobile devices has mushroomed. Just as location-based services grow fast, they lay out many questions and challenges when it comes to privacy. For example, who owns the location data and for what purpose is the data used? To answer these questions, we need new tools for location privacy. In this paper, we focus on the problem of secure location sharing, where a group of n clients want to collaborate with each other to anonymously share their location data with a location database server and execute queries based on them. To become more realistic, we assume up to a certain fraction of the clients are controlled arbitrarily by an active and computationally unbounded adversary. A relaxed version of this problem has already been studied in the literature assuming either a trusted third party or a weaker adversarial model. We alternatively propose a scalable fully-decentralized protocol for secure location sharing that tolerates up to $n/6$ statically-chosen malicious clients and does not require any trusted third party. We show that, unlike most other location-based services, our protocol is secure against traffic-analysis attacks. We also show that our protocol requires each client to send a polylogarithmic number of bits and compute a polylogarithmic number of operations (with respect to n) to query a point of interest based on its location.

1. INTRODUCTION

Nowadays, mobile users share their location data with other parties in order to receive information that is customized based on their location. For example, mobile users frequently ask *location-based services (LBS)* to find points of interest near them, to receive information about traffic along their route, or to receive customized advertising. On the other hand, shared location data can be used by others (e.g., providers and governments) for precise surveillance and hence, compromising user privacy.

FOMC'14, August 11, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2634274.2634281.

Although most organizations likely have good intentions, location information may be unintentionally leaked due to software bugs and vulnerabilities. The potential danger is that by compiling location patterns over time, untrusted parties could create an intimate picture of personal information like political and religious beliefs, health status, etc. The documents provided by Edward Snowden in 2013 show that NSA is collecting data about the locations of millions of cell phones by tapping into mobile networks [17]. Such data can be used to track the movements of individuals precisely and to find relationships between people by correlating various patterns in the locational data.

Due to the increasing concerns about location privacy, many governments and organizations have initiated studies on location privacy. For example, the U.S. government has recently initiated the discussion on the Location Privacy Protection Act [1], which is aimed at forcing companies to receive user consent before collecting mobile location information.

So far, most privacy-preserving LBS have assumed the existence of a trusted third party [21, 16, 26, 3], which is often used for anonymizing location data. Unfortunately, finding such trusted parties in practice is usually very hard or even impossible. Due to the huge number of mobile users interested in sharing their location data frequently, such centralized parties should be powerful servers that are usually owned by large companies and governments. Moreover, centralized parties may be subject to governmental control, and may be banned or forced to disclose sensitive location information.

LBS Privacy Approaches. Main approaches for achieving location privacy can be divided into two categories that provide either *location confidentiality* or *location anonymity*. Location confidentiality is to hide[1] user's location and to process locational queries over hidden data. Such data are never revealed to any authority although it might be possible to link the user's identity to its hidden query. Location anonymity, on the other hand, is to hide the connection between a locational query and the user who issues the query. In anonymity terminology, this is often called *unlinkability*, which means that a particular message is not linkable to any sender (or recipient), and that to a particular sender (recipient), no message is linkable [28].

While location confidentiality provides a great level of privacy, processing queries over hidden (e.g., encrypted) data

[1]By hiding, we mean using techniques like encryption, steganography, and secret sharing for obscuring data.

is usually hard and expensive. For example, homomorphic encryption [19] and private information retrieval [11] can be used for privacy-preserving query processing but known such techniques are still computationally intensive. On the other hand, if the number of users having queries is large, which is often the case for LBS, then the queries can be anonymized efficiently to provide a strong level of privacy.

Adversarial Model. In designing location-based services, there is a need to ensure reliability even against the most powerful type of adversary, called *active adversary*. Such an adversary controls a certain fraction of the parties to run sophisticated active attacks like jamming, corruption, and forging, as well as simple passive attacks like eavesdropping and non-participation. In an age where governments and financial entities increasingly engage in sophisticated cyber-warfare, we believe protecting against active attacks is crucial.

Resistance to Traffic-Analysis. One challenging problem with most anonymity-based location services is resistance against *traffic-analysis*. A global adversary can sniff traffic exchanged between the user and the service provider to link a query containing location information to the user who has issued that query. Such a powerful adversary was assumed to be unrealistic in the past but it might be realistic today if the service provider is controlled or compromised by a state-level surveillance authority [15]. Unfortunately, most anonymity-providing protocols like Casper [26], Prive [20], and Tor [14] are not secure against traffic analysis attacks. Recently, Zamani et al. [31] proposed a provably-secure anonymous broadcast protocol that resists traffic-analysis attacks. Unfortunately, their protocol has polylogarithmic rounds of communication and is vulnerable to collisions common to DC-Net-based protocols [10].

k-Anonymous LBS. Several LBS are built upon a relaxed notion of anonymity called *k-anonymity* [21, 26, 20, 16], where the adversary is assumed to be unable to identify the actual sender/receiver of a locational query from a set of k parties (called *anonymity set*). Even though *k-anonymity* often increases efficiency significantly, choosing small k's can result in severe privacy problems. For example, attackers often have background knowledge and it is shown that small anonymity sets are likely to leak privacy when attackers have such knowledge [25]. We argue that this is often the case for location anonymity as queries issued by people in different locations usually contain side information about their location. For example, a person located in New Mexico is more likely to search for a restaurant serving chili stew than a person in Vermont. Thus, we believe an algorithm is needed that scales well with the size of anonymity set, k.

Location Cloaking. In a seminal work, Gruteser and Grunwald [21] introduced two key ideas for achieving location k-anonymity called *spatial cloaking* and *temporal cloaking*. In spatial cloaking, a client's location (e.g., a two-dimensional point) is converted into a spatial area such that there are k mobile clients located in the area. While a spatially-cloaked location can be calculated efficiently (e.g., using the techniques of [21, 16, 26, 20]), the inaccuracy associated with the cloaked location often results in sending unnecessary information back to the user. In temporal cloaking, location anonymity is achieved by delaying the user's query until k clients also issue their queries. One drawback of this method is increased latency, especially when k is large. On the other hand, with the fast growth of the number of mobile users sharing their location information, this latency problem is becoming less important.

1.1 Our Contribution

We design a decentralized protocol for private location sharing that is secure against active attacks including traffic-analysis. Our protocol is efficient and scales well with the number of clients. Moreover, our protocol is load-balanced meaning that each client handles a roughly equal amount of communication and computation. This is crucial for our model since mobile devices usually have limited resources. We use techniques from *multi-party computation (MPC)*, where a set of n parties, each having a secret value, compute a known function over their inputs, without revealing the inputs to any party.

In the last three decades, a large body of work has been devoted to designing MPC protocols [6, 5, 18, 12]. Unfortunately, most of these protocols are inefficient and scale poorly with the number of parties. Recently, Boyle et al. [7] and Dani et al. [13] proposed scalable solutions to general MPC. Unfortunately, neither of these protocols are practical due to large logarithmic and constant factors in their communication/computation costs. Moreover, the protocol of Boyle et al. is not load-balanced making it hard to be used in settings where the parties have limited resources like mobile networks. Inspired by [7] and [13], our main strategy for achieving scalability is to perform local communications and computations in logarithmic-size groups of parties called *quorum*s, where the number of adversary-controlled parties in each quorum is guaranteed not to exceed a certain fraction. Using quorums, we develop an efficient multi-party shuffling protocol for anonymizing client queries.

Our protocol is provably-secure as it is based on a formal security framework, which follows from the secrecy of MPC. We show that the anonymity achieved by this method is, in particular, resistant to traffic analysis. We also provide provable anonymity against *a priori* knowledge that an adversary might have regarding the potential communicating parties. Moreover, unlike the majority of previous work which rely on centralized trusted servers, our protocol is fully-decentralized and does not require any trusted party.

Protocol Overview. In our protocol, we use the temporal cloaking approach to wait until k clients each hold a locational query. The k clients, then, participate in a MPC protocol to jointly compute a shuffling function that randomly permutes their queries. The shuffled queries are then sent to the location-based server to be processed. Finally, the results are broadcast by the LBS to all participating clients. More specifically, our protocol builds a set of quorums in a one-time setup phase and then uses them in the online phase for shuffling client queries. We represent the desired shuffling function by an arithmetic circuit. We assign the computation of each gate of the circuit to a quorum and evaluate the circuit level by level, passing the outputs of one level as the inputs to the next level. Figure 1 shows our protocol architecture. Each circle depicts a quorum of mobile users who connect to their local base station. Once the local computation is finished in each quorum, the result is forwarded to the next quorum via one-to-one communica-

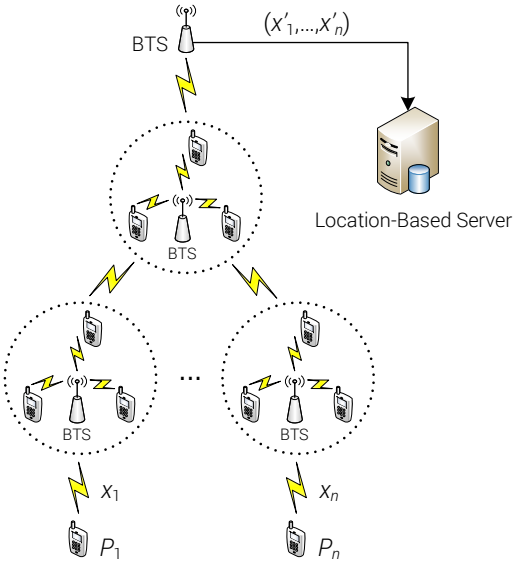

Figure 1: Our architecture

tion with clients of the next quorum. Finally, at the highest level, the shuffled queries are computed and sent to the LBS.

2. MODEL

We consider a network of n clients whose identities are common knowledge. We assume there is a private and authenticated communication channel between every pair of clients and the communication is *synchronous*. Our protocol does not require the presence of any trusted third party, and we do not assume the existence of a reliable broadcast channel. We assume $t < (1/6 - \epsilon)n$ of the clients are controlled by an *active* adversary, for some fixed, positive constant ϵ. We assume our adversary is *computationally unbounded* and is actively trying to prevent the protocol from succeeding by attacking the privacy of the parties, and the integrity of communications, by attempting to corrupt, forge, or drop messages. We say that the clients controlled by the adversary are *dishonest* and that the remaining clients are *honest* meaning that they strictly follow our protocol. We finally assume that the adversary is *static* meaning that it must select the set of dishonest clients at the start of the protocol.

3. OUR RESULTS

We provide a decentralized protocol for location-based services with polylogarithmic communication and computation costs with respect to the number of clients. We prove the following main theorem in Section 7.

THEOREM 1. *Consider n clients in a fully connected synchronous network with private channels, where each client has a locational query to send to a server. There exists an n-party protocol such that if all honest clients follow the protocol, then with high probability:*

- *Each honest client sends its query to the server anonymously and receives the result of query from the server.*
- *The protocol tolerates up to $t < (1/6 - \epsilon)n$ malicious clients.*
- *Each client sends $\tilde{O}(1)$ bits and computes $\tilde{O}(1)$ operations.*
- *The protocol has $O(\log^2 n)$ rounds of communication.*

4. RELATED WORK

Using anonymity for location privacy was first proposed by Kong and Hong [24]. They propose an anonymous routing protocol called ANDOR that targets mobile ad-hoc networks. They address two closely-related unlinkability problems, namely route anonymity and location privacy. Based on a route pseudonymity approach, ANODR prevents the adversary from exposing local wireless transmitters' identities and tracing network packet flows. For location privacy, their protocol ensures that the adversary cannot discover the real identities of local transmitters.

Gruteser and Grunwald [21] show that location data introduces new and potentially more severe privacy risks than network addresses pose in conventional services. Moreover, they analyzed the technical feasibility of k-anonymous location-based services and showed the privacy risks can be reduced through it. They propose an algorithm and service model that that can be used by a centralized location broker service and guarantee k-anonymous location information.

Zhong et al. [32] propose three protocols, Louis, Lester and Pierre, to achieve location privacy for a service that alerts a person of his nearby friend. Location privacy guarantees that users of the service can learn a friend's location if and only if the friend is actually nearby. Their approach was based on secure two-party computation and all three protocols exploit homomorphic encryption. The Louis protocol requires a semi-trusted third party that does not learn any location information. The Lester and Pierre protocols do not need a third party. The Lester protocol has the drawback that a user might be able to learn a friend's location even if the friend is in an area that is no longer considered nearby by the friend. The Pierre protocol does not have this disadvantage at the cost of not being able to tell the user the precise distance to a nearby friend.

The Casper framework of Mokbel et al. [26] consists of two main components called *location anonymizer* and *privacy-aware query processor*. The location anonymizer is a trusted third party that acts as a middle layer between mobile users and the location-based database server. It receives the location information, blurs the information into cloaked spatial areas, and sends the cloaked spatial areas to the location-based database server. The authors also design a privacy-aware query processor that helps database server to deal with anonymous queries and cloaked spatial areas rather than the exact location information. Unfortunately, the cloaked locations in Casper are usually very large and the algorithm lacks a mechanism to dynamically determine the size of cloaking regions.

Ghinita et al. [20] propose a decentralized model that helps mobile users self-organize into a fault-tolerant overlay network in order to run privacy-preserving anonymous location-based queries. Their protocol develops a rectangular area enclosing k users called *k-Anonymous Spatial Region (k-ASR)* in order to guarantee query anonymity even if the attacker knows the locations of all users. k-ASRs are built in a decentralized fashion, therefore the bottleneck of a centralized server is avoided. The empirical results confirm that

the system achieves efficient and scalable anonymization and load-balancing with low maintenance overhead, while being fault-tolerant. On the other hand, the protocol does not provide provable security guarantees and can only tolerate non-adversarial fail-stop faults.

Gedik and Liu [16] develop a k-anonymizer that is run by a trusted server. Their algorithm enables each mobile client to specify the minimum level of anonymity it desires and the maximum temporal and spatial tolerances it is willing to accept when requesting for k-anonymity. The authors propose a location cloaking algorithm called *CliqueCloak*, which combines the ideas of spatial and temporal cloaking. Unfortunately, the algorithm is expensive and shows poor performance for large k as it relies on the ability to locate a clique in a graph to perform location cloaking.

The PrivacyGrid framework of Bamba et al. [3] is composed of dynamic grid-based spatial cloaking algorithms for providing location k-anonymity and location ℓ-diversity (as defined in [25]) for mobile environments. The algorithms find the smallest possible cloaking region meeting desired privacy levels by measuring several metrics for k-anonymity and ℓ-diversity. While the algorithms are shown to be highly efficient and to achieve higher anonymization success rate, they rely on a trusted server for location tracking and anonymization service.

5. PRELIMINARIES

In this section, we define standard terms, notation, and results used throughout the paper.

5.1 Notation

An event occurs *with high probability*, if it occurs with probability at least $1-1/n^c$, for any $c > 0$ and all sufficiently large n. We denote the set of integers $\{1, ..., n\}$ by $[n]$. Also, let $\mathbb{Z}_p$ denote the additive group of integers modulo a prime p.

5.2 Basic Tools

In this section, we review the definitions of standard basic tools used throughout the paper.

Verifiable Secret Sharing. An (n, t)-*secret sharing* scheme, is a protocol in which a dealer who holds a secret value shares the secret among n parties such that any set of t parties cannot gain any information about the secret, but any set of at least $t+1$ parties can reconstructs it. An (n, t)-*verifiable secret sharing (VSS)* scheme is an (n, t)-secret sharing scheme with the additional property that after the sharing phase, a dishonest dealer is either disqualified or the honest parties can reconstruct the secret, even if shares sent by dishonest parties are spurious. In our protocol, we use the constant-round VSS protocol of Katz et al. [22] that is based on Shamir's secret sharing scheme [29]. This result is described in Theorem 2.

Theorem 2. [22] *There exists a synchronous linear (n, t)-VSS scheme for $t < n/3$ that is perfectly-secure against a static active adversary. The protocol requires one broadcast and three rounds of communication.*

Quorum Building. A *good quorum* is a set of $N = O(\log n)$ parties that contains a majority of honest parties. King et al. [23] showed that a *Byzantine Agreement (BA)* protocol can be used to bring all parties to agreement on a collection of n good quorums. In this paper, we use the fast BA protocol of Braud-Santoni et al. [8] to build n good quorums.

Theorem 3. [23, 8] *There exists an unconditionally-secure protocol that brings all good parties to agreement on n good quorums with high probability. The protocol has $\tilde{O}(n)$ amortized communication and computation complexity[2], and it can tolerate up to $t < (1/3 - \epsilon)n$ malicious parties.*

Secure Broadcast. In the malicious setting, when parties have only access to secure pairwise channels, a protocol is required to ensure secure (reliable) broadcast[3]. Such a broadcast protocol guarantees all parties receive the same message even if the broadcaster (dealer) is dishonest and sends different messages to different parties. It is known that a BA protocol can be used to perform secure broadcasts. In our protocol, we use the BA algorithm of Braud-Santoni et al. [8] to perform secure broadcasts.

Theorem 4. [8] *There exists an unconditionally-secure protocol for performing secure broadcasts among n parties. The protocol has $\tilde{O}(n)$ amortized communication and computation complexity, and it can tolerate up to $t < (1/3 - \epsilon)n$ malicious parties.*

Sorting Networks. A *sorting network* is a network of *comparators*. Each comparator is a gate with two input wires and two output wires. When two values enter a comparator, it outputs the lower value on the top output wire, and the higher value on the bottom output wire. Ajtai et al. [2] describe an asymptotically-optimal $O(\log n)$ depth sorting network called *AKS*. Unfortunately, the AKS network is not practical due to large constants hidden in the depth complexity. Batcher [4] constructs a simple and efficient sorting circuit with depth $1/2 \log n(1+\log n) = O(\log^2 n)$. The idea of Batcher's circuit is to sort $n = m_1 + m_2$ elements by sorting the first m_1 and the last m_2 independently, and then applying a (m_1, m_2)-*merging network* to the result. In our protocol, we use the this Batcher's circuit for shuffling client queries efficiently.

Secure Comparison. Given two linearly secret-shared values $a, b \in \mathbb{Z}_p$, Nishide and Ohta [27] propose an efficient protocol for computing a sharing of $\rho \in \{0, 1\}$ such that $\rho = (a \leq b)$. Their protocol has $O(1)$ rounds and requires $O(\ell)$ invocations of a secure multiplication protocol, where ℓ is the bit-length of elements to be compared. We refer to this protocol by Compare. We also describe a fast multiplication protocol to be used along with the comparison protocol of [27] for implementing fast comparator gates.

Share Renewal. In our protocol, a shuffling circuit is securely evaluated. Each gate of the circuit is assigned a quorum Q and the parties in Q are responsible for comparison of secret-shared inputs. Then, they send the secret-shared result to any quorums associated with gates that need this result as input. Let Q' be one such quorum. In order to secret-share the result to Q' without revealing any information to any individual party (or to any coalition of dishonest

[2]Amortized communication complexity is the total number of bits exchanged divided by the number of parties.

[3]We are not aware of any easy approach to physically implement a broadcast channel without assuming a trusted party.

parties), a fresh sharing of the result must be distributed in Q'. To this end, we use the share renewal protocol of Zamani et al. [30]. Combined with the VSS scheme of Theorem 2 in a group of n parties with $t < n/3$ dishonest parties, this protocol has only one round of communication and requires each party to send $O(n)$ field elements. We refer to this protocol by RenewShares throughout our protocol.

6. OUR PROTOCOL

In this section, we describe our protocol for secure location sharing. Consider n parties $P_1, P_2, ..., P_n$ each having a locational query $x_i \in \mathbb{Z}_p$, for a prime p and all $i \in [n]$. The parties want to anonymously send their queries to a location-based server (LBS) and receive the results back. We assume the queries have already been collected via temporal cloaking and up to $t < (1/6 - \epsilon)n$ of the parties are controlled by a malicious adversary.

We first describe an ideal functionality of our protocol where a hypothetical trusted party P computes the desired protocol outcome by communicating with all parties[4]. In every run of the protocol, P executes a shuffling protocol over the queries and sends the shuffled sequence of queries to the location-based server. Once the queries are processed, the server broadcasts the results to the parties. The shuffling protocol first chooses a uniform random number $r_i \in \mathbb{Z}_p$ to form an input pair (r_i, x_i) for each party P_i and for all $i \in [n]$. The protocol then uses a shuffling circuit that is based on the Batcher's sorting network [4] to sort the set of pairs $\{(r_1, x_1), ..., (r_n, x_n)\}$ with respect to their first elements (Figure 2). We later show that this functionality randomly permutes the set of inputs $\{x_1, ..., x_n\}$.

In our protocol, we denote the shuffling circuit by C, which has m gates. Each gate is essentially a comparator gate and is denoted by G_i, for $i \in [m]$. Our protocol first creates n quorums and then assigns each gate of C to a quorum. For n parties, the circuit has $\lceil n/2 \rceil$ input gates as each gate has two inputs. We label the quorums assigned to the input gates by $Q_1, ..., Q_{\lceil n/2 \rceil}$ and call them *input quorums*. C also has $\lceil n/2 \rceil$ output gates, which correspond to *output quorums* labeled by $Q'_1, ..., Q'_{\lceil n/2 \rceil}$.

We now implement the real functionality of our protocol based on the ideal functionality described above. Protocol 1 defines our main algorithm. Throughout the protocol, we represent each shared value $s \in \mathbb{Z}_p$ by $\langle s \rangle = (s_1, ..., s_n)$ meaning that each party P_i holds a share s_i generated by the VSS scheme of Theorem 2 during its sharing phase. Using the natural component-wise addition of representations, we define $\langle a \rangle + \langle b \rangle = \langle a + b \rangle$.

For multiplication, we define $\langle a \rangle \cdot \langle b \rangle = \mathsf{Multiply}(\langle a \rangle, \langle b \rangle)$, where Multiply is the multiplication protocol of Zamani et al. [30]. This protocol is based on a well-known technique proposed by Beaver [5] and generates a shared multiplication triple $(\langle u \rangle, \langle v \rangle, \langle w \rangle)$ such that $w = u \cdot v$. The triple is then used to convert multiplications over shared values to additions. The first step of this protocol is independent of the inputs and thus, for efficiency purposes, it can be performed in an offline phase to generate a sufficient number of multiplication triples.

In our protocol, we use a simple and well-known technique (as described by Beaver [5]) for generating uniformly random secrets by adding uniformly random secrets shared by each party. We refer to this algorithm by GenRand.

[4]We are inspired by the standard ideal/real world definition for multi-party protocols proposed by Canetti [9].

7. PROOFS

In this section, we prove the correctness and secrecy of Theorem 1. First, in Lemma 2, we show that for sufficiently large $k > 0$ and $q > \frac{3}{2}kn^2 \log n$, LocationSharing computes a random permutation of the input queries with high probability.

DEFINITION 1. ***[Perfect Random Permutation]*** *Consider a set of $n \geq 1$ elements $A = \{a_1, a_2, ...a_n\}$. A perfect random permutation of A is a permutation chosen uniformly at random from the set of all possible $n!$ permutations of A.*

LEMMA 1. *Consider a sequence of input pairs (r_1, x_1),..., (r_n, x_n), and a sorting protocol Π that sorts the pairs according to their first elements. Π computes a perfect random permutation of the pairs if their first elements are chosen uniformly at random and are distinct.*

PROOF. Let $X = (r_1, x_1), ..., (r_n, x_n)$ be the input sequence and $Y = (s_1, y_1), ..., (s_n, y_n)$ be the output sequence of Π. Note $s_1, ..., s_n$is the sorted sequence of $\{r_1, ..., r_n\}$. An arbitrary output sequence of pairs $Y' = (s'_1, y'_1), ..., (s'_n, y'_n)$ is said to be equal to Y if $y_i = y'_i$, for all $i \in [n]$. We want to prove that the probability of Y' being equal to Y is $\frac{1}{n!}$. In general, for any $i \in [n]$, $y_i = y'_i$ if and only if s'_i is the i-th smallest element in $\{r_1, ..., r_n\}$ conditioned on knowing the $i-1$ smallest elements, which happens with probability $\frac{1}{n-i+1}$. Thus, the probability that $Y = Y'$ is $\frac{1}{n} \cdot \frac{1}{n-1} \cdot ... \cdot \frac{1}{2} \cdot 1 = \frac{1}{n!}$. □

In the random generation step of LocationSharing, it is possible that two or more input quorums choose the same random elements from $\mathbb{Z}_q$. In this situation, we say a *collision* happens. Collisions reduce the level of anonymity our protocol guarantees because the higher the probability of collisions, the higher the chance the adversary is given in guessing the correct sequence of inputs. On the other hand, we observe that if the field size (i.e., q) is sufficiently large, then the probability of collisions becomes overwhelmingly small. Lemma 2 gives a lower bound on q such that collisions are guaranteed to happen with negligible probability.

LEMMA 2. *Let $\mathbb{Z}_q$ be the field of random elements generated in the random generation step of* LocationSharing. *The probability that there is a collision between any two parties is negligible if $q > \frac{3}{2}kn^2 \log n$, for some $k > 0$.*

PROOF. Based on Theorem 3, all input quorums are good. Based on the correctness of GenRand, all elements generated by the input quorums in the random generation step of LocationSharing are chosen uniformly at random and independent of all other random elements generated throughout the protocol. Let P_i and P_j be two parties and r_i and r_j be the random values assigned to them respectively by their corresponding input quorums. The probability that $r_i = r_j$ is $1/q$. Let X_{ij} be the following indicator random variable and Y be a random variable giving the number of collisions between any two parties,

$$X_{ij} = \begin{cases} 1, & r_i = r_j \\ 0, & otherwise \end{cases}, \quad Y = \sum_{i,j \in [n]} X_{ij}.$$

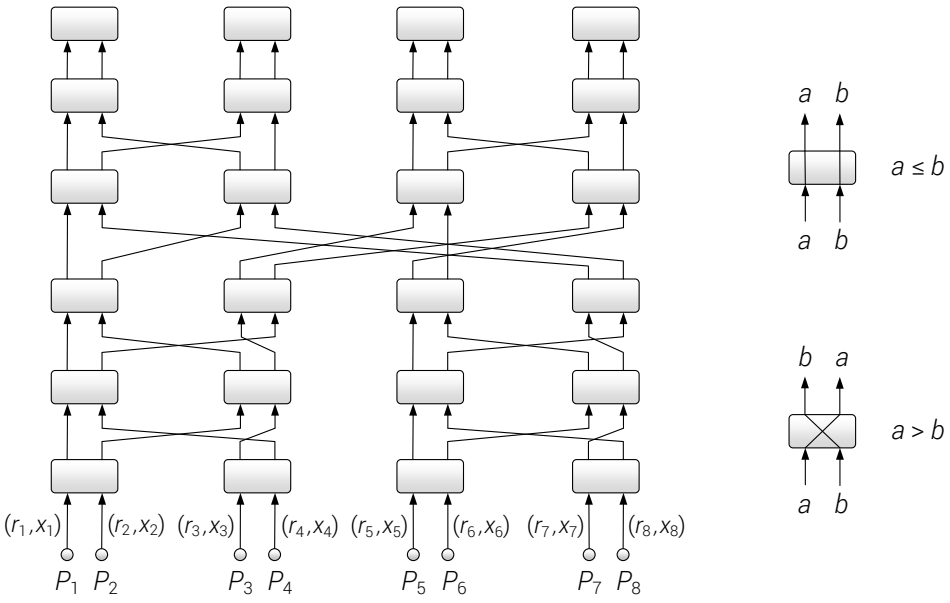

Figure 2: The shuffling circuit for $n = 8$ (left), and the behavior of a comparator gate (right)

Using the linearity of expectations,

$$E(Y) = E\Big(\sum_{i,j\in[n]} X_{ij}\Big) = \sum_{i,j\in[n]} E(X_{ij}) = \frac{1}{q}\binom{n}{2} = \frac{n(n-1)}{2q}.$$

We want to find an upper bound on the probability of collisions using the Chernoff bound defined by

$$Pr(Y \geq (1+\alpha)E(Y)) \leq e^{-\frac{\alpha^2 E(Y)}{3}}.$$

To ensure that no collision happens with high probability, we need to have $(1+\alpha)E(Y) < 1$ while $e^{-\frac{\alpha^2 E(Y)}{3}} < \frac{1}{n^k}$, for any $k > 0$. Choosing $\alpha < \frac{1}{E(Y)} - 1$ and solving the inequalities for $E(Y)$ we get

$$e^{-\frac{\alpha^2 E(Y)}{3}} < \frac{1}{n^k} \quad \Rightarrow \quad e^{-\frac{\alpha^2 E(Y)}{3}} < e^{-k\log n} \quad \Rightarrow$$
$$1 - \frac{\alpha^2 E(Y)}{3} < -k\log n \quad \Rightarrow \quad \frac{(\alpha+1)^2 E(Y)}{3} > k\log n \quad \Rightarrow$$
$$\frac{1}{3E(Y)} > k\log n \quad \Rightarrow \quad E(Y) < \frac{1}{3k\log n}.$$

Since $E(Y) = \frac{n(n-1)}{2q} < \frac{1}{3k\log n}$, solving this for q gives the bound $q > \frac{3}{2}kn^2\log n$ and $\alpha < 3k\log n - 1$. □

7.1 Proof of Theorem 1

We prove in the real/ideal world model as described by Canetti [9]. First, we consider the protocol in an ideal model. In this model, all parties send their input to a trusted party who computes the shuffling circuit. Then, it sends the result to the location-based server. Let A be the sequence of inputs and A' be the sequence of sorted inputs according to the random numbers associated with them. Recall that we have at least $n - t$ honest parties. Based on Lemma 1 and conditioned on the event that no collision happens with high probability (Lemma 2), the elements of A' that correspond to honest parties can be any permutation of the elements of honest parties in A. In other words, the probability that the adversary can successfully map A' to A is less than $\frac{1}{(n-t)!}$ which guarantees $(n-t)$-anonymity (i.e., full anonymity). Protocol LocationSharing is the realization of the above ideal model. The real model, computes the circuit in a multi-party setting. We prove this realization is correct and secure.

Setup. The correctness and secrecy follows from the proof of Theorem 3.

Input Broadcast. The correctness and secrecy follows from the proof of the VSS scheme of [22]. After this step, each input quorum Q_i has a correct sharing of P_i's input. This is the base case for our proof of circuit computation step.

Random Generation. The correctness and secrecy follows from the proof of the GenRand algorithm.

Circuit Computation. *Correctness.* We prove by induction in the real/ideal model. The invariant is that if the input shares are correct, then the output of each gate is equal to the output when the gate is computed by a trusted party in the ideal model, and the result is shared between parties of the quorum correctly. For the base case, note that the invariant is true for input gates. Induction step is based on the universal composability of Compare and Multiply. Moreover, based on the correctness of RenewShares, the output resharing step generates new shares without changing the output.

Secrecy. We prove by induction. The adversary cannot obtain any information about the inputs and outputs during the computation of each gate of the circuit. Let Q, and Q' be two quorums involved in the computation of a gate, where Q provides an input to the gate, and Q' computes the gate. Consider a party P. Let S be the set of all shares P receives during the protocol. We consider two cases. First, if $P \notin (Q \cup Q')$, then elements of S are independent of the shares Q sends to Q'. Moreover, elements of S are independent of the output of Q' since Q' also re-shares its output(s).

Protocol 1 LocationSharing

1. **Setup:** Parties run the quorum building protocol of Theorem 3 jointly with all parties to agree on n good quorums $Q_1, ..., Q_n$, and assign each gate G_i to the (i mod n)-th quorum. For each quorum Q, each party in Q runs the key generation protocol of VSS scheme of Theorem 2 jointly with other parties in Q.

2. **Input Broadcast:** Parties P_i and P_{i+1} secret share their inputs x_i and x_{i+1} among all parties of $Q_{\lceil i/2 \rceil}$.

3. **Random Generation:** Parties in each input quorum $Q_{\lceil i/2 \rceil}$ run GenRand twice to generate sharings $\langle r_i \rangle$ and $\langle r_{i+1} \rangle$ of two random elements $r_i, r_{i+1} \in \mathbb{Z}_q$, for some prime q. At the end of this step, $Q_{\lceil i/2 \rceil}$ holds two pairs of sharings $(\langle r_i \rangle, \langle x_i \rangle)$ and $(\langle r_{i+1} \rangle, \langle x_{i+1} \rangle)$.

4. **Circuit Computation:** The circuit is evaluated level-by-level starting from the input gates. For each gate G and the quorum Q associated with it, parties in Q do the following.

   (a) **Gate Computation:** Let $(\langle r \rangle, \langle x \rangle)$ and $(\langle r' \rangle, \langle x' \rangle)$ be the sharings associated with the inputs of G. Parties compute $\langle \rho \rangle = \mathsf{Compare}(\langle r \rangle, \langle r' \rangle)$, where $\rho = (r \leq r')$. Then, they compute the output pairs $(\langle s \rangle, \langle y \rangle)$ and $(\langle s' \rangle, \langle y' \rangle)$ from

   $$\langle s \rangle = \langle \rho \rangle \cdot \langle r \rangle + (1 - \langle \rho \rangle) \cdot \langle r' \rangle$$
   $$\langle y \rangle = \langle \rho \rangle \cdot \langle x \rangle + (1 - \langle \rho \rangle) \cdot \langle x' \rangle$$
   $$\langle s' \rangle = \langle \rho \rangle \cdot \langle r' \rangle + (1 - \langle \rho \rangle) \cdot \langle r \rangle$$
   $$\langle y' \rangle = \langle \rho \rangle \cdot \langle x' \rangle + (1 - \langle \rho \rangle) \cdot \langle x \rangle$$

   For every addition over two shared values $\langle a \rangle$ and $\langle b \rangle$ performed above, parties computes $\langle c \rangle = \langle a \rangle + \langle b \rangle$. For every multiplication, they run $\langle c \rangle = \mathsf{Multiply}(\langle a \rangle, \langle b \rangle)$.

   (b) **Output Resharing:** Parties run RenewShares over $\langle s \rangle$, $\langle y \rangle$, $\langle s' \rangle$, and $\langle y' \rangle$ to re-share them in the quorum associated with the parent gate.

5. **Query Processing:** Let $(\langle s_i \rangle, \langle y_i \rangle)$ and $(\langle s_{i+1} \rangle, \langle y_{i+1} \rangle)$ be the pairs of shared values each output quorum Q'_i receives. Parties in Q'_i run $z_i = \mathsf{Reconst}(\langle y_i \rangle)$ and $z_{i+1} = \mathsf{Reconst}(\langle y_{i+1} \rangle)$ and send z_i and z_{i+1} to the LBS. For all the z_i and z_{i+1} messages received from parties of Q'_i, the LBS picks two via majority filtering. Then, the LBS processes the queries $z_1, ..., z_n$, and sends the results to all parties.

Hence, S reveals nothing about the inputs and outputs of the gate.

Second, if $P \in (Q \cup Q')$, then the inductive invariant is that the collection of all shares held by dishonest parties in Q and Q' does not give the adversary any information about the inputs and the outputs. As the base case, it is clear that the invariant is valid for input gates. The induction step is as follows. The adversary can obtain at most $2(N/6) = N/3$ shares of any shared value during the computation step; $N/6$ from dishonest parties in Q and $N/6$ from dishonest parties in Q'. By the secrecy of the VSS scheme, at least $N/3+1$ shares are required for reconstructing the secret. By the secrecy of RenewShares and Multiply, when at most $N/3$ of the shares are revealed, the secrecy of the computation step is proved using universal computability of multi-party protocols.

Query Processing. The correctness and secrecy follows from the proof of the Reconst algorithm and the standard assumption that the location-based server correctly executes the queries and sends the results back to the parties.

LEMMA 3. *The protocol* LocationSharing *sends* $\tilde{O}(n)$ *bits, computes* $\tilde{O}(n)$ *operations, and has* $O(\log^2 n)$ *rounds of communication.*

We first compute the cost of each step of the protocol separately and then compute the total cost. Based on [22], the communication and computation complexity of the VSS subprotocol is $O(\mathsf{poly}(n))$ when it is invoked among n parties. Also, the VSS protocol takes constant rounds of communication.

- *Setup.* The communication and computation costs are equal to those costs of the quorum building algorithm of Theorem 3, which is $\tilde{O}(1)$ for each party. This protocol takes constant rounds of communication.
- *Input Broadcast.* The input broadcast step invokes the VSS protocol n times among $N = O(\log n)$ parties. So, this step sends $O(n \cdot \mathsf{poly}(N))$ bits and performs $O(n \cdot \mathsf{poly}(N))$ operations. Since the VSS scheme is constant-round, this step also takes constant rounds.
- *Random Generation.* It is easy to see that the subprotocol GenRand sends $O(N \cdot \mathsf{poly}(N))$ messages, performs $O(N \cdot \mathsf{poly}(N))$, and has constant rounds.
- *Circuit Computation.* The Batcher's sorting network [4] has $O(n \log^2 n)$ gates. So, the communication cost of this step is equal to the communication and computation cost of running $O(n \log^2 n)$ instantiations of Compare and RenewShares. Compare requires $O(\log q)$ invocation of Multiply which sends $O(\mathsf{poly}(N))$ messages and computes $O(\mathsf{poly}(N))$ operations. RenewShares also sends $O(\mathsf{poly}(N))$ messages and computes $O(\mathsf{poly}(N))$ operations. Hence, the circuit computation phase sends $O(n \log n \cdot \log q \cdot \mathsf{poly}(N))$ messages computes $O(n \log n \cdot \log q \cdot \mathsf{poly}(N))$. Since the sorting network has depth $O(\log^2 n)$, and Compare takes constant rounds, this steps takes $O(\log^2 n)$ rounds of communication.
- *Query Processing.* The costs are equal to the communication and computation costs of running n instantiations of Reconst. Thus, this step sends $\tilde{O}(n)$ messages and performs $\tilde{O}(n)$ operations. Since Reconst

is a constant-round protocol and the communications with the server take constant rounds, this step of the protocol also takes constant rounds.

- *Total.* Since $q > \frac{3}{2}kn^2 \log n$, for a constant k, $q = O(n^3)$ and $\log q = O(\log n)$. Thus, Protocol 1 sends $\tilde{O}(n)$ bits and computes $\tilde{O}(n)$ operations. Finally, the protocol requires $O(\log^2 n)$ rounds of communication. This completes the proof of Theorem 1.

8. CONCLUSION AND OPEN PROBLEMS

We described a secure location sharing protocol that is fully decentralized and tolerates up to $n/6$ active faults. Moreover, our protocol is load-balanced and can tolerate traffic-analysis attacks. The amount of information sent and the amount of computations performed by each client scales polylogarithmically with the number of parties. The scalability is achieved by performing local communications and computations in groups of logarithmic size and by relaxing the latency requirements.

Several open problems remain. For example, can we design a spatial cloaking technique to trade-off anonymity for efficiency while preserving the traffic-analysis resistance? Or, can we decrease the number of rounds of our protocol using a smaller-depth sorting circuit? For example, since our protocol sorts uniform random numbers, it seems possible to use a logarithmic depth non-comparison-based sorting circuit like bucket sort to achieve $O(n)$-anonymity.

9. ACKNOWLEDGMENTS

The authors would like to acknowledge supports from the National Science Foundation (NSF) under grant number CCF-1320994. The authors would also like to thank Jared Saia from University of New Mexico and Aniket Kate from Saarland University for their valuable contributions to the discussions and for their supportive comments.

10. REFERENCES

[1] S.1223 - Location Privacy Protection Act of 2012.

[2] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3(1):1–19, January 1983.

[3] Bhuvan Bamba, Ling Liu, Peter Pesti, and Ting Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 237–246, New York, NY, USA, 2008. ACM.

[4] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), pages 307–314, New York, NY, USA, 1968. ACM.

[5] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer Berlin Heidelberg, 1991.

[6] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computing. In *Proceedings of the Twentieth ACM Symposium on the Theory of Computing (STOC)*, pages 1–10, 1988.

[7] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation: how to run sublinear algorithms in a distributed setting. In *Proceedings of the 10th theory of cryptography conference on Theory of Cryptography*, TCC'13, pages 356–376, Berlin, Heidelberg, 2013. Springer-Verlag.

[8] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast Byzantine agreement. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 57–64, New York, NY, USA, 2013. ACM.

[9] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, FOCS '01, pages 136–145, Oct 2001.

[10] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[11] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, November 1998.

[12] I. Damgård, Y. Ishai, M. Krøigaard, J. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. *Advances in Cryptology – CRYPTO '08*, pages 241–261, 2008.

[13] Varsha Dani, Valerie King, Mahnush Movahedi, and Jared Saia. Quorums quicken queries: Efficient asynchronous secure multiparty computation. In *Distributed Computing and Networking*, volume 8314 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg, 2014.

[14] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[15] Joan Feigenbaum and Bryan Ford. Seeking anonymity in an Internet panopticon. *e-Print arXiv:1312.5307*, March 2014.

[16] B. Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 620–629, June 2005.

[17] Barton Gellman and Ashkan Soltani. NSA tracking cellphone locations worldwide, snowden documents show. The Washington Post, December 2013.

[18] R. Gennaro, M.O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing*, PODC '98, pages 101–111. ACM, 1998.

[19] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[20] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Prive: Anonymous location-based queries in distributed mobile systems. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 371–380, New York, NY, USA, 2007. ACM.

[21] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 31–42, New York, NY, USA, 2003. ACM.

[22] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of vss in point-to-point networks. In Luca Aceto, Ivan Damgård, LeslieAnn Goldberg, MagnusM. Halldorsson, Anna Ingolfsdottir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, volume 5126 of *Lecture Notes in Computer Science*, pages 499–510. Springer Berlin Heidelberg, 2008.

[23] Valerie King, Steven Lonargan, Jared Saia, and Amitabh Trehan. Load balanced scalable Byzantine agreement through quorum building with full information. In MarcosK. Aguilera, Haifeng Yu, NitinH. Vaidya, Vikram Srinivasan, and RomitRoy Choudhury, editors, *Distributed Computing and Networking*, volume 6522 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg, 2011.

[24] Jiejun Kong and Xiaoyan Hong. Anodr: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '03, pages 291–302, New York, NY, USA, 2003. ACM.

[25] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. on Knowledge Discovery from Data*, 1(1), March 2007.

[26] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: Query processing for location services without compromising privacy. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB '06, pages 763–774. VLDB Endowment, 2006.

[27] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 343–360. Springer Berlin Heidelberg, 2007.

[28] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity – a proposal for terminology. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9. Springer Berlin Heidelberg, 2001.

[29] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[30] Mahdi Zamani, Mahnush Movahedi, and Jared Saia. Millions of millionaires: Multiparty computation in large networks. Cryptology ePrint Archive, Report 2014/149, 2014.

[31] Mahdi Zamani, Jared Saia, Mahnush Movahedi, and Joud Khoury. Towards provably-secure scalable anonymous broadcast. In *the 3rd USENIX Workshop on Free and Open Communications on the Internet*, FOCI '13, 2013.

[32] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, PET'07, pages 62–76, Berlin, Heidelberg, 2007. Springer-Verlag.