P3: Privacy Preserving Positioning for Smart Automotive Systems

SIAM UMAR HUSSAIN and FARINAZ KOUSHANFAR, University of California San Diego

This article presents the first privacy-preserving localization method based on provably secure primitives for smart automotive systems. Using this method, a car that is lost due to unavailability of GPS can compute its location with assistance from three nearby cars, while the locations of all the participating cars including the lost car remain private. Technological enhancement of modern vehicles, especially in navigation and communication, necessitates parallel enhancement in security and privacy. Previous approaches to maintaining user location privacy suffered from one or more of the following drawbacks: trade-off between accuracy and privacy, one-sided privacy, and the need of a trusted third party that presents a single point to attack. The localization method presented here is one of the very first location-based services that eliminates all these drawbacks. Two protocols for computing the location is presented here based on two Secure Function Evaluation (SFE) techniques that allow multiple parties to jointly evaluate a function on inputs that are encrypted to maintain privacy. The first one is based on the two-party protocol named Yao's Garbled Circuit (GC). The second one is based on the Beaver-Micali-Rogaway (BMR) protocol that allows inputs from more than two parties. The two secure localization protocols exhibit trade-offs between performance and resilience against collusion. Along with devising the protocols, we design and optimize netlists for the functions required for location computation by leveraging conventional logic synthesis tools with custom libraries optimized for SFE. Proof-of-concept implementation of the protocol shows that the complete operation can be performed within only 355ms. The fast computing time enables localization of even moving cars.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**;

Additional Key Words and Phrases: Connected cars, secure automotive system, location privacy, location-based services, secure function evaluation, garbled circuit

ACM Reference format:
Siam Umar Hussain and Farinaz Koushanfar. 2018. P3: Privacy Preserving Positioning for Smart Automotive Systems. *ACM Trans. Des. Autom. Electron. Syst.* 23, 6, Article 79 (November 2018), 19 pages.
https://doi.org/10.1145/3236625

1 INTRODUCTION

Contemporary automobiles are increasingly being equipped with advanced technologies that make significant enhancements to both functionality and safety of the vehicles. Two of the most significant improvements in this field are smart navigation system and inter-vehicle

This work is supported in parts by an Office of Naval Research grant (ONR-R17460), National Science Foundation grants (CNS-1059416 and CCF-1116858), Semiconductor Research Corporation grant (2013-HJ-2471) and Multidisciplinary Research Program of the University Research Initiative grant (FA9550-14-1-0351).

Authors' addresses: S. U. Hussain and F. Koushanfar, Department of ECE, University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093; emails: {siamumar, farinaz}@ucsd.edu.

1084-4309/2018/11-ART79 $15.00
https://doi.org/10.1145/3236625

communication, facilitating sharing of important information like traffic update, environmental hazards, accidents, or road work. A large class of modern vehicle also includes an intra-network of processors connected to a central CPU providing Ethernet, USB, Bluetooth, and IEEE 802.11 interfaces [20]. Besides enhancing performance, these technologies also create new dimensions for attack. Thus, in addition to classic vehicular reliability requirement, security and privacy of the user should be taken into careful consideration while implanting these advanced features [9, 20, 31]. Moreover, due to the increasing reliance on these smart features, backup plans to cope with the failure of one or more components is also crucial for reliability.

In this article, we present the first privacy-preserving localization method for smart cars based on provably secure primitives. With this method, a car lost due to unavailability of GPS can send requests to three nearby cars to get assistance in finding its location. The three assisting cars then engage in a privacy-preserving triangle localization protocol to estimate the location of the lost car. The locations of all the cars including the lost car remain private.

To date, the most widely explored method to ensure user privacy in Location-Based Services (LBS) is location cloaking [10, 16, 24]. In this method, instead of sending the exact location and time instant of the user, a range of area covered in a period of time is sent. To make sure that the user's location cannot be inferred from this data, the range and period are chosen such that there are at least $k - 1$ other users in that area during that period, which ensures "k-anonymity" of the user. k-anonymity requires the existence of a trusted third party called anonymizer that combines the user location with locations of other users subscribed to the service. This anonymizer presents a single point to attack the system. Moreover, cloaking is also vulnerable to context-based attack and trajectory-tracing. More importantly, the approximate location results in noisy and stochastic response to the query. While this approximate response may be acceptable in some LBS scenario, for localization and navigation applications the accuracy of the method is crucial.

The work in References [12, 25, 45] explored performing the location-based query (e.g., nearest neighbor) in a transformed space. These methods increase the accuracy over the cloaking approaches. However, they still have few drawbacks. For example, Reference [25] proposed three methods that either requires a semi-trusted third party or has to sacrifice accuracy or privacy for simplified operation. The authors in References [12, 25] consider the privacy of only one party (client), while the data of the other party (server) is assumed to be public.

To compute accurate results while maintaining complete privacy of all the participating parties, we design two protocols employing two Secure Function Evaluation (SFE) techniques: Yao's Garbled Circuit (GC) [41] and Beaver-Micali-Rogaway (BMR) [3]. Yao's GC is currently considered to be the most effective provable privacy-preserving technique [8, 18]. This protocol allows two parties to jointly evaluate a function on inputs that are encrypted to maintain privacy. The BMR protocol is a variant of GC that supports more than two parties. Unlike the previous methods, neither of GC or BMR protocols involve trade-off between accuracy and privacy. To date application of SFE in LBS has been limited. The work in Reference [1] presents application-specific solutions to some simple problems like point-inclusion, intersection, and closest pair based on GC. The work in Reference [19] presents an implementation of the nearest neighbor query with GC. These methods require sharing encryption keys with another party, which poses a security threat. Our work is the first practical privacy-preserving location-based application that employs SFE techniques effectively and securely.

We devise two protocols where three cars assist in estimating the location of the lost car. The protocols are based on the secure computation of the triangle localization algorithm presented in Reference [35]. In the first protocol, the three assisting cars participate in a total six invocations of the two-party GC operation such that the locations of all cars including the lost car remain private. To cope with the time constraint due to car movement, the protocol is designed such that each car can simultaneously participate in two GC operations with each of the two other cars (assuming a multi-core architecture of the processors, which is widely available at present). With this protocol, the location of the lost car is secure as long as at least one of the assisting cars does not collude with the other cars. The second protocol involves only one invocation of the multi-party BMR operation. This protocol is secure against collusion among any number of cars. However, the BMR protocol requires more computation than the GC and thus the second protocol is more time consuming than the first one.

In both GC and BMR, the pertinent function is represented as a circuit consisting of Boolean logic gates (AND, OR, XOR, etc). This circuits is called a *netlist*. We generate the netlists required for the localization protocol by using conventional logic synthesis tools with GC optimized custom libraries as suggested in Reference [39]. Even though some of the optimizations for GC are not available for BMR, the netlist optimization goal still remains the same. Therefore, the methodology in Reference [39] can be employed to generate netlists for BMR with slight modification. Our custom synthesis library includes the first GC (and BMR) optimized implementations of division and square-root functions, required for the computation of the location of the lost car. The synthesis library presented in Reference [39] include implementations of unsigned addition, subtraction, and multiplication. We add enhanced implementations of these functions to our library to support signed inputs and overflow.

One major use case for our privacy-preserving localization is in military applications when a lost military vehicle requires help in locating itself. It is crucial that the location of each participating vehicle remain private so that an adversarial vehicle cannot learn their location by pretending to be an ally or by tapping into the common channel. This application can also be beneficial in verifying a suspected vehicle claimed location via distance bounding with assist from three nearby cars. Generally, three verifying base stations perform distance bounding on the suspect vehicle confining it to a triangular region. However, this requires costly infrastructure that may not be available in all places. In this scenario, three other cars can act as the verifying base stations while their locations remain private and the location of the suspect vehicle is revealed only to the verifier.

Contributions: In brief, our contributions are as follow:

- We present the first privacy-preserving triangle localization for smart automotive systems based on provably secure primitives. We design two protocols utilizing SFE techniques such that a lost car along with three nearby cars can jointly compute the location of the lost car while the locations of all the participating cars remain private.
- We analyze the security and performance of the localization protocols in detail and demonstrate the trade-off between performance and collusion deterrence.
- We develop a circuit synthesis library with functions required to generate GC and BMR optimized netlists for triangle localization algorithm. This library includes the first GC implementations of square-root and division operations.
- Proof-of-concept implementation of our protocol demonstrates practicality of the design. The complete protocol is performed within only 355ms.

Note that an earlier version of this work was presented in Reference [21], which included only the GC-based protocol. In this version, we provide a detailed analysis on the collusion among the participating cars and show that in the GC-based protocol the location of the lost car is secure as long as at least one of the three assisting cars do not collude with others. We then present a second protocol based on BMR, which supports more than two participants as opposed to GC. This protocol is secure even if all the assisting cares collude with one another. In addition to this, we optimize the netlist for computation of location by determining the maximum physical bound

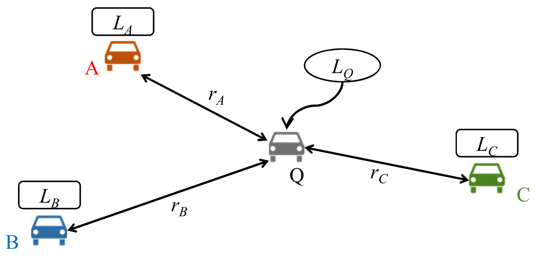


Fig. 1. Overview of the localization algorithm.

on the intermediate variables. As a result, the end-to-end localization time is reduced from 550 to 355ms on the same evaluation platform.

The rest of this article is organized as follows. In the next section, we present a generic overview of the localization protocol. In Section 3, we briefly describe the background required to explain this work. Next, we outline the protocol and analyze the privacy of the cars in Section 4. In Section 5, we describe the details of the SFE operation, and in Section 6, we evaluate our implementation of the protocol in terms of time and resource usage. In Section 7, we survey the related literature. Finally, Section 8 concludes the paper.

2 GLOBAL FLOW

The overview of the localization process is displayed in Figure 1. The lost car Q sends requests to three nearby cars A, B, and C to assist in computing its location. The first step is to estimate the distance r_X of Q from each assisting car X (= A, B, or C). Depending on the protocol used, either the assisting car or the lost car learns this distance, but not both of them. The location L_X of each car X is known only to itself throughout the protocol. Then A, B, C, and Q (only in the second protocol) participate in a privacy-preserving localization protocol to compute the location of Q.

Ideally, the location of Q would be a common intersection of three circles centered at A, B, and C. However, due to inaccuracy in distance estimation, the location of Q is computed as the median of a triangle formed by the intersections of pairs of circles. In the first protocol, each pair of cars (say, A and B) participates in a GC operation to compute two possible candidates for one vertex of the triangle. Then one of them (say, B) participates in another GC operation with the third car (C) to select the candidate closer to C as the vertex. Thus, six GC operations are required to determine all three vertices of the triangle. One car can learn zero to at most two vertices. Therefore, a single car cannot compute the median on its own. The median L_Q, i.e., the location of Q, is computed through *secure sum* [11] protocol where all four cars participate and revealed only to Q. The second protocol employs BMR, which supports more than two (in this case four) participants. In this one, the complete operation, including the computation of the median, is performed with only one invocation of the SFE protocol. Therefore, the intermediate values (intersecting points) are not revealed to any participant, making it secure against collusion.

Security Model. Consistent with the earlier relevant literature [1, 12, 19, 25, 38, 45], we adopt the *honest-but-curious* security model [4, 27]. In this model, the participating parties follow the agreed-upon protocol but may want to learn about the other parties' data from the information at hand (their own input and the output received from the protocol). Moreover, for privacy-preserving protocols involving more than two parties, there is the notion of *honest majority*, where the number of honest parties is higher than the number of dishonest parties. Of the two localization protocols presented in this article, the first one requires an honest majority. However, honest majority is not a requirement for the second localization protocol.

3 PRELIMINARIES

In this section, we provide a brief background related to this work. We first explain the cryptographic tools we employ to ensure privacy while computing the location. Especially, we describe the SFE frameworks used for secure computation. Finally, we describe the localization algorithm used to compute the location of the lost car.

3.1 Cryptographic Protocols

Oblivious Transfer. Oblivious Transfer (OT) [29] is a cryptographic protocol executed between a sender S and a receiver R, where R selects one from a pair of messages provided by S without revealing her selection. In a 1-out-of-2 OT protocol, (OT_1^2), S holds a pair of messages (m_0, m_1); R holds a selection bit $b \in 0, 1$ and obtains m_b without revealing b to S and learns nothing about m_{1-b}.

Yao's Garbled Circuit. Yao's Garbled Circuit (GC) [41] is a cryptographic protocol where two parties, Alice and Bob, jointly compute a function $z = f(x_a, x_b)$ on their private inputs x_a, provided by Alice, and x_b, provided by Bob. In the end, one or both of them learn the output z. The function f is represented as a Boolean circuit, called *netlist*, consisting of 2-input 1-output logic gates. Thus, the operation of each gate is described by a four-entry truth-table. The netlist is *simulated* in a way such that the actual value of each wire is shared between Alice and Bob, and none of them can learn the value individually. Alice, called the *garbler*, garbles the circuit as follows. She assigns each wire in the netlist with a pair of 128-bit random keys corresponding to the Boolean values 1 and 0. For each gate, a garbled truth table is constructed by encrypting the keys for output with the corresponding input keys. She then sends the garbled circuit along with the keys corresponding to her input values (one key per input) to Bob, called the *evaluator*. Bob obtains the keys corresponding to his input values obliviously through 1-out-of-2 OT protocol that allows him to retrieve the keys without revealing the values of his inputs. He then uses these input keys to evaluate the encrypted tables gate by gate and decrypt the keys associated with the value of each wire. However, the mapping of these keys to the actual values is known only to Alice. Thus, together they share the secret value of each wire. At the final step, they reveal their respective shares for only the output wires to learn the output z.

Beaver-Micali-Rogaway. BMR [3] is a multi-party variant of Yao's GC supporting more than two parties. In contrast to GC, where only one party generates the garbled circuit, all the parties jointly participate in the preparation of the garbled circuit, and no subset of colluding parties can learn any value internal to the netlist. The function is of the form $z = f(x_0, x_1, \ldots, x_{n-1})$, where there are n parties involved and x_i is the private input of the ith party. Each of the n parties assigns each wire in the netlist with a pair of 128-bit random keys corresponding to the values 1 and 0. To construct the garbled truth table, the keys of the output wire of each gate is encrypted separately with corresponding input keys from each party. Thus, the keys of the input wire from just one party is able to hide the keys of the output wire. However, it comes with the increased computation cost of $O(n^2)$ and communication cost of $O(n)$ per gate, instead of $O(1)$ in GC. The construction of all the garbled gates are independent and is performed in parallel. Evaluation phase is performed only by the parties receiving outputs. For each corresponding input wire, each party sends the keys generated by them to each evaluating party and each evaluating party receives the keys generated by other parties through OT. Each evaluating party then evaluates the garbled circuit gate by gate. The evaluation requires $O(n^2)$ decryption operation per gate as opposed to $O(1)$ in GC. As a result, this protocol is more time consuming compared to GC.

Optimizations and Netlist Generation. A number of optimizations to the GC protocol have been proposed: free-XOR [26], row reduction [30], half gate [42], and fixed-key block cipher [4].

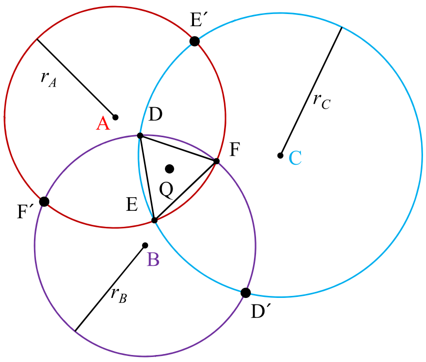


Fig. 2. Triangle localization algorithm. The lost car is Q and the assisting cars are A, B, and C. The calculated location of Q is the centroid of the triangle DEF.

Among these optimizations, a major one is free-XOR, as it allows the evaluation of `XOR`, `XNOR`, and `NOT` gates (jointly termed as the `XOR` gates) without costly cryptographic encryption, which also translates to less communication time as the `XOR` gates do not need the transfer of the garbled tables. As a result, the primary optimization goal while generating the netlist for f is to minimize the number of non-`XOR` gates (`AND`, `OR`, `NAND`, etc). The size of the non-`XOR` gate truth table is reduced by 25% each with the row-reduction and half gate optimizations. Among these, only the free-XOR optimization is compatible with BMR [5].

According to the methodology presented in TinyGarble [39], f is first described with a Hardware Description Language (HDL) and compiled with a logic synthesis tool using libraries that are designed to minimize the number of non-`XOR` gates. As a result, it naturally benefits from the powerful techniques already incorporated in existing logic synthesis solutions.

Note that the realizations of oblivious transfer, Yao's GC, and BMR protocol employed in this work are secure in the honest-but-curious security model (with honest majority for the BMR protocol). Realizations of oblivious transfer and Yao's GC in the *malicious* security models, where the parties may deviate from the agreed-upon protocols, has been presented by recent works. However, these realizations incur more computational cost to ensure security in this stronger setting. Since the localization protocols presented in this article are only secure in the honest-but-curious model, incurring the extra cost of those realizations would be redundant.

3.2 Triangle Localization

Figure 2 shows the setup of the triangle localization algorithm provided in Reference [35]. The car Q is lost. It requests three other cars, A, B, and C, to help locate itself.

First, distances r_A, r_B, and r_C of Q from A, B, and C, respectively, are estimated. In the ideal case where the estimated distance is exactly equal to the actual distance, the three circles centered at A, B, and C with radii r_A, r_B, and r_C, respectively, would have a common intersection at Q. However, in practice distance cannot be estimated so precisely. An underestimation may result in no intersection. Therefore, the distance is generally overestimated. In this way, a triangle DEF is formed by the points of intersections. The estimated location of Q is the median of the triangle.

In general, two circles intersect at two points (for example, circles with centers at A and B intersect at F and F'). The one that falls inside the third circle forms one vertex of the triangle (F falls inside the circle centered at C). The equations for calculating the coordinates of F and F' is provided here. The other intersections can be calculated in similar fashion. We denote the

Euclidean coordinates of a point P as (x_P, y_P).

$$\sqrt{(x_F - x_A)^2 + (y_F - y_A)^2} = r_A, \tag{1}$$

$$\sqrt{(x_F - x_B)^2 + (y_F - y_B)^2} = r_B, \tag{2}$$

$$\sqrt{(x_F - x_C)^2 + (y_F - y_C)^2} \leqslant r_C, \tag{3}$$

$$x_F = \frac{1}{2p}(y_F q + t), \tag{4}$$

$$y_F = \frac{1}{p^2 + q^2}\left(pqx_A + y_B p^2 - \frac{1}{2}qt \pm \frac{1}{2}\sqrt{(qt - 2y_A p^2 - 2pqx_A)^2 - s(p^2 + q^2)}\right), \tag{5}$$

$$\begin{aligned} where,\ & p = x_B - x_A,\ q = y_B - y_A, \\ & t = r_A^2 - r_B^2 + x_B^2 - x_A^2 + y_B^2 - y_A^2, \\ & s = (4p^2 y_A^2 + t^2 - 4ptx_A + 4p^2 x_A^2 - 4p^2 r_A^2). \end{aligned}$$

Equations (1) and (2) have two solutions as given by Equations (4) and (5). The one that lies inside the range of C, decided through inequality Equation (3), forms one vertex of the triangle. Note that the vertex of the triangle is shown as F in the figure just for simplicity; it could be either of F or F'.

4 PROTOCOL AND ANALYSIS

We designed two protocols to securely compute the location of the lost car. The first one is based on the two-party SFE protocol, Yao's GC. We break down the localization function into six invocations of the GC protocol between the three assisting cars. With this protocol, the location of the lost car is secure as long as at least one of the assisting cars does not collude. The second protocol is based on the multi-party SFE protocol, BMR. This protocol is simpler and more secure than the first one, as all four cars participate in one invocation of the BMR protocol. The computed location remains secure even if all three lost cars collude with one another. However, this protocol takes four times longer to compute the location as compared to the first one.

In the following, we describe the two protocols and analyze the security and privacy of them. The lost car is denoted as Q and the three assisting cars are denoted as A, B, and C. We then analyze the privacy of the location of the participating cars without and with collusion among the cars.

4.1 Protocol with Yao's GC

4.1.1 Protocol Description. There are two phases in this protocol. In the first phase, the coordinates of the triangle DEF are computed through the GC protocol. For the location verification scenario, the coordinates are provided to the verifying authority after this phase. For other localization scenarios, the median of the triangle is computed through the *Secure Sum* [11] protocol in the second phase.

Phase 1: Computing Triangle *DEF*. For this phase, we need to evaluate the netlists of following two functions through GC. Similar to the previous section, the computation of the vertex F is used as an example here.

$[x_F, y_F, x'_F, y'_F] = Intersection(x_A, y_A, r_A, x_B, y_B, r_B)$,
which implements Equations (4) and (5).

$in_F = Range(x_F, y_F, x_C, y_C, r_C)$,
which implements inequality Equation (3).

The steps of this phase are as follows.

(i) Q sends LOCK_LOCATION request to A.
(ii) Upon receiving the request, A locks its current coordinates (x_A, y_A) and immediately start the estimation of the distance r_A with Q.
(iii) Steps (i) and (ii) are repeated with B and C where they lock their respective coordinates (x_B, y_B) and (x_C, y_C) immediately prior to the start of distance estimation. The estimated distances with B and C are denoted as r_B and r_C, respectively.
(iv) A and B compute the coordinates $F(x_F, y_F)$ and $F'(x'_F, y'_F)$ of the intersections of their circles by evaluating the *Intersection* netlist through Yao's GC protocol. The output map is configured such that A learns $F(x_F, y_F)$ and B learns $F'(x'_F, y'_F)$.
(v) B and C jointly decide whether F' lies inside the range of C by evaluating the *Range* netlist through Yao's GC protocol. The output in_F is 1 if F' lies inside the range of C, and 0 otherwise, in which case the intersection F lies inside the range of C. B learns in_F and shares it with A. C learns nothing in this step.
(vi) B and C perform the Step (iv). B learns $D(x_D, y_D)$ and C learns $D'(x'_D, y'_D)$.
(vii) C and A perform the Step (v) to compute in_D, which is 1 if D' lies inside the range of A or 0 if D lies inside the range of A. C learns in_D and shares it with B. A learns nothing in this step.
(viii) C and A perform the Step (iv). C learns $E(x_E, y_E)$ and A learns $E'(x'_E, y'_E)$.
(ix) A and B perform the Step (v) to compute in_E, which is 1 if E' lies inside the range of B or 0 if E lies inside the range of B. A learns in_E and shares it with C. B learns nothing in this step.

Phase 2: Computing the Median of Triangle *DEF*. After phase 1, each assisting car possesses the coordinates of two intersections and two Boolean variables indicating whether or not these intersections are vertices of the triangle DEF. In this phase, the assisting cars along with the lost car Q compute the median of the triangle through the following steps:

(i) Q sends a random coordinate (x, y) to A.
(ii) A computes the sums $X_A = (x + \overline{in}_F.x_F + in_E.x'_E)$ and $Y_A = (y + \overline{in}_F.y_F + in_E.y'_E)$ and sends to B.
(iii) B computes the sums $X_B = (X_A + \overline{in}_D.x_D + in_F.x'_F)$ and $Y_B = (Y_A + \overline{in}_D.y_D + in_F.y'_F)$ and sends to C.
(iv) C computes the sums $X_C = (X_B + \overline{in}_E.x_E + in_D.x'_D)$ and $Y_C = (Y_B + \overline{in}_E.y_E + in_D.y'_D)$ and sends to Q.
(v) Q now subtracts the initial random numbers from the sums and compute the medians as $((X_C - x)/3,\ (Y_C - y)/3)$, which are the coordinates of its estimated location.

4.1.2 Security Analysis. We now analyze what information each car can learn regarding the location of the other cars.

Lost Car. In this protocol, the lost car learns nothing but its own location. However, there is a maximum range within which the cars will be able to communicate with each other. If that range is R, then the lost car can assume that the three assisting cars are within a circular area around it with a radius of R. Therefore, the uncertainty over the location of the assisting cars is $1/\pi R^2$.

Assisting Cars. An assisting car can be interested in two types of information: the locations of the other two assisting cars and the location of the lost car. Each assisting car knows the coordinates of

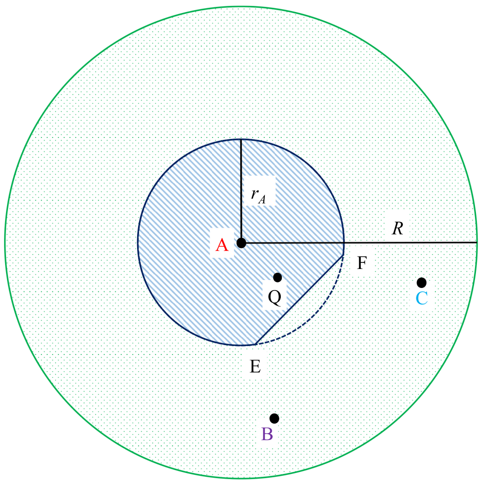


Fig. 3. The regions of uncertainty for car A in locating the other cars. The uncertainty region of the lost car Q is marked with stripes and the uncertainty region of the other two assisting cars B and C is marked with dots.

only one of the intersections with the circle of the other two assisting cars. Without the coordinates of the other intersection, it is not possible to deduce the center of the other circle. Therefore, the uncertainty for one assisting cars over the location of other two assisting cars is $1/\pi R^2$.

Regarding the location of the lost car, an assisting car knows the distance between the lost car and itself with some uncertainty created by the lost car by modifying the propagation time as described later in Section 4.4. Therefore, an assisting car X (= A or B or C) can confine the location of the lost car within a circular region with radius r_X. It is possible for one assisting car to know the coordinates of two of the vertices of the triangle DEF. Those two vertices form one chord of that circle. In a strict sense, it is not possible to learn which side of that chord the other vertex resides. However, if the two partitions on either side of the chord have largely different areas, it is more likely that the other vertex is on the larger partition. Even though it is not straightforward to calculate the uncertainty here, the minimum uncertainty, in this case, would be $2/\pi r_X^2$.

The regions of uncertainty for car A in locating the other cars is shown in Figure 3. The uncertainty region of the lost car Q is marked with stripes and the uncertainty region of the other two assisting cars B and C is marked with dots. It is assumed that A knows the vertices E and F of DEF.

Collusion Among Cars. In this protocol, the lost car Q does not participate in any invocation of the SFE protocol. Intuitively, if all three assisting cars collude with one another the location of the lost car will not remain secure. Indeed after Step (iii) of Phase 1, the cars A, B, and C collectively know all the inputs to the Equations (3), (4), and (5). Therefore, together they can compute the location of the lost car. Another point to note here is that based on the relative location of Q, there is a possibility that one of the three assisting cars learns two vertices of the triangle while one other car knows none of them. In that case, it would be enough for two cars to collude to compute the location of the lost car. However, it is not possible to predict this scenario before the start of Phase 2.

4.2 Protocol with BMR

4.2.1 Protocol Description. The possible security breach in the previous protocol arises due to two fact that the lost car holds no inputs to the secure function. Since Yao's GC allows only two inputs, to involve the lost car in the secure computation, we would have to break down both the

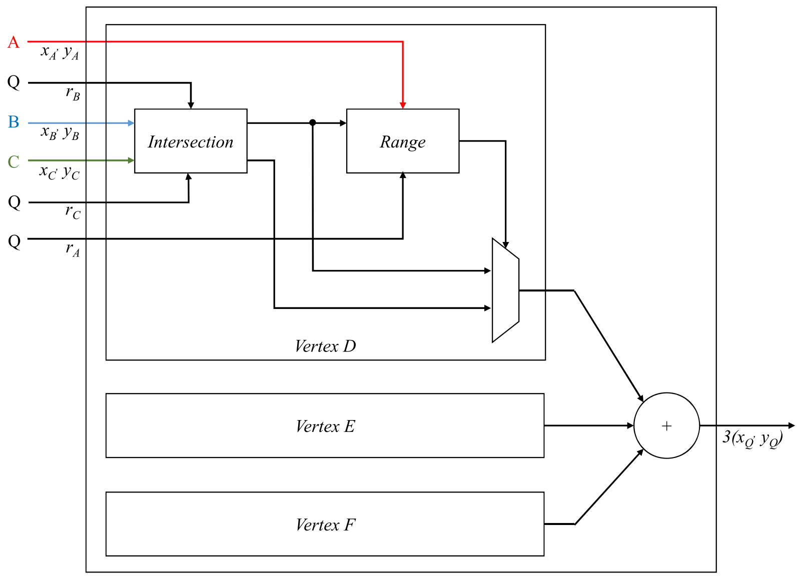


Fig. 4. The *TriLoc* netlist to compute the location of the lost car Q with help from three assisting cars A, B, and C through the BMR protocol. Only the netlist for computing the vertex D is shown in detail.

Intersection and *Range* functions into two parts each and perform twelve GC operations instead of six. However, we present another protocol based on BMR that supports inputs from more than two parties.

This protocol involves only one invocation of the BMR operation where all four parties participate. It requires only one netlist that includes three instances each of the *Intersection* and *Range* netlists. The netlist, named *TriLoc*, is outlined in Figure 4. Only the netlist for computing vertex D is shown in detail. Unlike the first three steps of Phase 1 in the previous protocol, the distances r_A, r_B, r_C of Q, respectively, with A, B, and C are estimated by Q (the coordinates of A, B, and C are still locked by the respective cars). Therefore, Q now holds three inputs to the Equations (3), (4), and (5). All of A, B, C, and Q performs garbling operation, while only Q evaluates the netlist and thus learns the output. In location verification scenario, the output is revealed to the verifier instead of Q.

4.2.2 Security Analysis. The analysis on the regions of uncertainty for this protocol is similar to the first one. Since the lost car is the one estimating the distances instead of the assisting cars, their respective regions of uncertainty also switch. The lost car now can confine the three assisting cars A, B, and C within circular regions with radii r_A, r_B, r_C, respectively, and the assisting cars can confine the lost cars within circular regions with radius R. The regions of uncertainty of the assisting cars with respect to one another remains the same.

Collusion Among Cars. As explained above, the location of the lost car is secure with this protocol even if all three of the assisting cars collude. However, unlike the previous protocol, there is a possibility of collusion between the lost car and one or more of the assisting cars. If, say, C colludes with Q, then together they hold the information regarding the distances of A and C from Q: r_A and r_C, respectively. The maximum distance between A and C is $r_A + r_C$. If this distance is shorter than the maximum communication distance R, then C can confine the location of A within a distance of $r_A + r_C < R$, which will result in shrinking the region of uncertainty. Since, in this protocol, the intersections between the circles are internal variables of the secure function, as shown in Figure 4, the location of A cannot be predicted with an accuracy more than this.

4.3 Effect of the Motion of Cars

The inputs to the two functions *Intersection* and *Range* are locked in the first three steps of Phase 1 of the protocol. The rest of the protocol execution proceeds with these locked values. Therefore, the final output of the protocol revealed to Q is the location of Q at the end of these three steps. There are two timing constraints that affect the accuracy of the estimated location:

(1) The time to lock the coordinates of A, B, and C and estimating the distances should be negligibly small such that all the cars can be considered stationary during that time period. As shown in References [32, 33] the distance estimation can be done as fast as in a few nanoseconds. Therefore, the time in the first three steps primarily consists of the times to send the LOCK_LOCATION request, which is only a few bits. According to our experimentation sending a 32-bit integer takes around 1,500 clock cycles, which translates to around 1.5μs. Therefore, the total time for these steps is around 3μs (note that the time for the LOCK_LOCATION request to the first car A does not need to be considered, since the process starts only after A receives that request). Assuming the cars are moving at 100kph, they move about 83μm in this period, which is indeed negligibly small.

(2) The total time of the protocol execution should be small enough so that the estimated location is close to the current location of Q. Another possibility is that Q remains stationary during the protocol execution. Note that the assisting cars do not need to be stationary, since their locations are locked at the beginning. As we show in Section 6, the time to complete the protocol is 330ms. Assuming the lost car is moving at 100kph, it will move about 9.3m during this period. Note that the current minimum accuracy of GPS coordinating systems is 8m [40].

4.4 Distance Compensation

According to the first protocol described in the previous section, one assisting car may know two vertices of the triangle DEF. The estimated location of Q is the median of DEF and is calculated through the secure sum protocol such that only Q learns the final result. However, if the area of the triangle is too small, the location of Q may be estimated by a car with good accuracy from just two vertices of DEF. To prevent this, Q should be allowed to manipulate the area of DEF by controlling the estimated distances from the three assisting cars. However, the estimated distance should only be known to the respective assisting car.

Among several methods available for distance estimation like Received Signal Strength Indicator [13, 43, 44], Time of Arrival [6, 13, 17], and Angle of Arrival [28, 34], the one most suitable for this purpose is the two-way Time of Arrival method [6].

In this method, the assisting car sends a synchronization message to the lost car and the lost car sends it back after some delay. Then, the assisting car measures the time shift (t_s) between the transmitted and received messages and subtract the estimated delay t_d to get the propagation time $t_p = t_s - t_d$. In a typical application, the delay accounts for the time to receive the complete the message, and the time for the transceivers of both the cars to change their mode (transmitter $\leftrightarrow$ receiver). In this application, the lost car can wait an arbitrary time before sending back the message so that the actual delay is larger than the estimated delay t_d. This increases the estimated distance and eventually results in a larger area of DEF.

Note that since the final location is the median of the triangle, the larger area does not result in a significant error in the estimated location as we will show in Section 6.

5 SFE OPERATION

As explained in Section 3.1, we need to generate a netlist consisting of Boolean logic with the optimization goal set to minimizing the number of non-XOR gates, which minimizes both communication and computation [26], to securely evaluate a function. In this section, we describe the generation of these netlists with our custom synthesis library and the invocation of the SFE protocols to securely evaluate these netlists.

5.1 Netlist Generation

We follow the TinyGarble methodology [39] to generate the netlists for GC and BMR operations. Even though TinyGarble supports both sequential and combinational circuits, the latter approach is more suited for the localization application as it does not involve repeated operation for most of the parts. The TinyGarble framework provides free-XOR optimized synthesis library that contains implementations of arithmetic functions like unsigned addition, subtraction, and multiplication. For implementations of Equations (3)–(5), we extend the library by including signed versions of these functions along with support for variable bit-length and overflow, which are essential for generating the netlist for any arbitrary practical function. In addition to this, we implemented free-XOR optimized division and square-root functions as required by Equations (4) and (5).

As shown in Figure 4, the netlist for $TriLoc$, required by the BMR-based protocol, is composed of the netlists for *intersection* and *range* functions, along with three MUXs and one three input adder. In the following, we discuss the generation of GC/BMR optimized netlists for these functions. The netlists for each function need to be generated only once. It is generated offline and saved in each car's memory.

Intersection. The $Intersection$ netlist computes Equations (4) and (5) that require, along with other arithmetic functions, division and square root. In our implementation, the complexity of the number of non-XOR gates in a w-bit division operation is $O(w^2)$, which is similar to the complexity of the multiplication operation provided in Reference [39]. The number of non-XOR gates for a 64-bit division operation is 12,546. The square-root operation follows an iterative procedure. The complexity of the number of non-XOR gates in a w-bit square-root operation with v iterations is $O(w^2 v)$. Again, the number of required iterations can be assumed to be linearly proportional to the bit width, which simplifies the term to $O(w^3)$. Therefore, the of the number of non-XOR gates in the $Intersection$ netlist with W-bit location coordinates is $O(W^3)$. The number of non-XOR gates for a 64-bit square-root operation with 32 iterations is 12,733.

If we start with W-bit Euclidean coordinates, then the number of bits in the internal variables keeps increasing due to overflow. The outputs of a w-bit addition/subtraction, multiplication, and division operations need $w + 1$, $2w$, and w bits, respectively. Going this way, inputs to the two division operations of Equation 5 is $3W + 7$-bit (note the "±" in the equation, hence two division operations). However, the output of this equation is the Euclidean coordinates of an intersection and at the boundary condition, these coordinates can be at most four times the highest possible coordinate of an assisting car. Therefore, the outputs of these division operations will be confined to the lowest $W + 2$ bits, and we can discard the rest. A similar situation occurs for the division operations for Equation 4. Besides reducing the number of non-XOR gates in the $Intersection$ netlist, this also reduces the number of non-XOR gates in the *Range* netlist as these coordinates are its inputs.

Range. Even though inequality Equation (3) involves a square-root operation, both sides of this inequality are positive quantities as both of them are measured distances. Therefore, we can avoid the costly square-root operation by squaring both sides. As a result, the *Range* netlist is much smaller than the $Intersection$ netlist, the most complex operation being squaring (multiplication) with a complexity of $O(w^2)$.

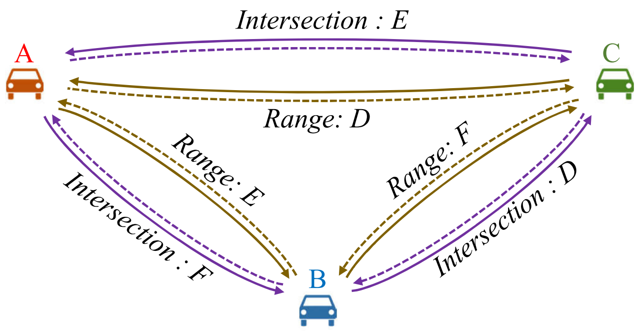


Fig. 5. Illustration of parallel invocations of GC protocol.

5.2 Invocation of the SFE Protocols

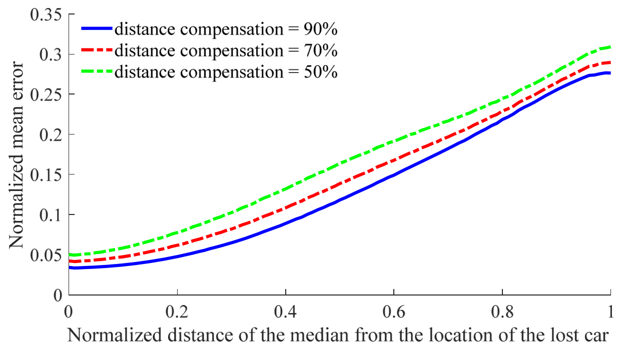
GC Operation. Each of the assisting cars participates in two GC operations on the *Intersection* netlist with the other two cars in the first protocol. These two GC operations are independent of each other and performed in parallel in two cores of the processor. To ensure symmetry, each car performs as the garbler for one pair and the evaluator for the other. Similarly, each assisting car participates in two parallel GC operations on the *Range* netlist with the other two cars. Figure 5 illustrates these operations. The outer arrows depict GC on *Intersection* and the inner arrows depict GC on *Range*. The vertex of the triangle DEF that is being computed in each GC operation is also indicated beside the arrows. A solid arrow emanating from a car indicates that the car acts as the garbler in that operation, and a dashed arrow indicates the evaluator.

The operation of the car A is described here as an example. A acts as the garbler while B acts as the evaluator to determine the coordinates of F and F' through the *Intersection* netlist and only learns the coordinate of F. In parallel to this, A participates in another GC operation as the evaluator, with C as the garbler to compute the coordinates of E and E' and learns only the coordinate of E'. A then performs as the garbler, while B performs as the evaluator to decide whether E' forms one vertex of the triangle through the *Range* netlist and shares the result with C. At the same time, it acts as the evaluator in another GC operation where C is the garbler to decide whether D' forms one vertex of the triangle without learning the result.

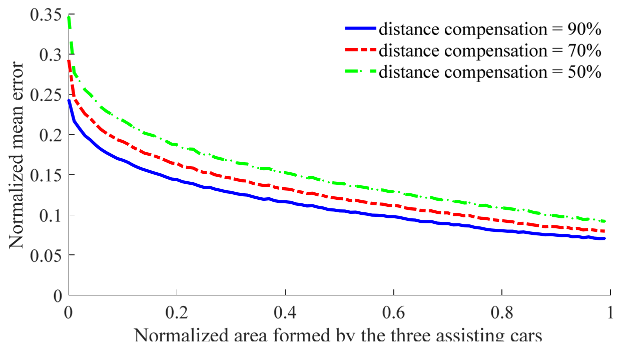
BMR Operation. With BMR the complete operation is performed in one invocation of the protocol on the *TriLoc* netlist. Even though the computation of each vertex is independent of each other as can be seen from Figure 4, BMR computes the circuit serially gate by gate. Therefore, the BMR-based protocol cannot benefit from the parallelism of the operations. Moreover, as explained in Section 3.1, The BMR protocol incurs computation cost of $\mathcal{O}(n^2)$ and communication cost of $\mathcal{O}(n)$, as opposed to $\mathcal{O}(1)$ in GC. The total number of computed gates also increases slightly, since the median computation is performed through SFE instead of the secure sum as in the GC-based protocol. As a result, while this protocol shows better resilience against collusion, it is slower than the first one. All of A, B, C, and Q act as garblers, while only Q acts as the evaluator and learns the final output, which is its location. Unlike the GC-based protocol, the intermediate results, i.e., the coordinates of the intersections, are not revealed to any car.

6 EVALUATION

In this section, we first analyze the error in location measurement associated with the triangle localization algorithm. Next, we evaluate the netlists required for the secure localization



(a) Normalized mean error in the estimated location of the lost car as a function of the normalized distance between the actual location of the lost car and the median of the triangle *ABC* with different degrees of distance compensation.



(b) Normalized mean error in the estimated location of the lost car as a function of the normalized area of the triangle *ABC* with different degrees of distance compensation.

Fig. 6. Error analysis.

protocol in terms of the number of non-XOR gates. Finally, we garble/evaluate them through the SFE frameworks and present the timing results.

6.1 Error Analysis

We first analyze the error in the location estimated by triangle localization algorithm. Note that this error is solely due to the localization method, and distance estimation error. The SFE protocols do not introduce any additional error. To estimate the error, we run simulation by placing the assisting cars at random positions inside a square area with dimension T and place the lost car at the center of that square. The error is quantified as the Euclidean distance between the estimated and actual location of the lost car, normalized to T. The estimation error depends on two factors: (a) the relative positions of the assisting cars with respect to the lost car, (b) the area of the triangle formed by the three assisting cars.

In Figure 6(a) the error is plotted against the distance (normalized to T) between the actual location of the lost car and the median of the triangle formed by cars A, B, and C. For each point on the curves, the simulation is run for $5.7E + 03$ times. The plot shows that the estimation error increases linearly with the relative distance between the location of the lost car and the triangle

Table 1. Number of XOR and Non-XOR Gates in the Netlists

Netlist	No. of non-XOR gates	No. of XOR gates	Total no. of gates
Intersection	2.40E + 04	6.71E + 04	9.11E + 04
Range	4.51E + 02	7.54E + 02	1.21E + 03
TriLoc	7.38E + 04	2.06E + 05	2.80E + 05

ABC. To analyze the effect of distance compensation, we simulate three cases where the actual distance is increased by 50%, 70%, and 90%, respectively. The plot shows that the estimation errors are fairly close for all three cases.

In Figure 6(b) the error is plotted against the area (normalized to T^2) of the triangle formed by cars A, B, and C. For each point on the curves, the simulation is run for $2E + 4$ times. The plot shows that the estimation error is high when the area is small, i.e, when the three assisting cars lie close to a straight line. The error decreases sharply with increase in the area. Similar to the previous case, distance compensation does not have a significant effect on the estimation error.

In cases where there are more than three assisting cars are available, it would be beneficial to choose the set of three cars that will result in the highest accuracy. Choosing the set according to the relative location of the assisting cars with respect to the lost car is not feasible, since it requires the knowledge about the location of the lost car. However, it is possible to compute the area formed by three cars and compare it against a predetermined threshold. To ensure privacy this computation is performed by the BMR protocol.

6.2 Circuit Synthesis

As explained in Section 3.1, to compute a function securely through the Yao's GC or the BMR protocol, the function needs to be represented as a netlist of Boolean logic gates. Three netlists are required for the SFE operations—*Intersection* and *Range* for GC and *TriLoc* for BMR. The equations for the first two netlists (Equations (4), (5), and (3)) are described using Verilog HDL and compiled with the Synopsys Design Compiler [22] with our custom libraries. The *TriLoc* netlist is constructed from the first two. Due to the free-XOR optimization, the XOR gates can be computed locally without costly cryptographic encryption or communication. Therefore, the total time to compute the function is determined solely by the number of non-XOR gates in the netlist. The number of non-XOR and XOR gates in the three netlists are presented in Table 1. It shows that the number of non-XOR gates are around only one-quarter of the total number of gates. This demonstrates the effectiveness of our customized synthesis library in generating the SFE-optimized netlist.

6.3 Timing Analysis

To assess the timing performance, we run the two localization protocols on a system with Ubuntu 14.10 Desktop, 12.0GB of memory, and Intel Core i7-2600 CPU @ 3.4GHz. We employ the TinyGarble framework [37] to perform the GC operations. The number of clock cycles in every phase of the GC operation to garble/evaluate the *Intersection* and *Range* netlists once is presented in Table 2. In the first localization protocol, each of these netlists is garbled/evaluated three times by the three assisting cars in parallel. The total number of clock cycles from the lost car initiating the operation to the final computation of its location is $1.20E + 09$, which translates to only 355ms. However, as described in Section 3.1, the input values to the functions are not required during the garbling operation. They are only required at the start of the oblivious transfer phase. Thereof ore, one way to reduce the accuracy loss due to the movement of the lost car is to lock the coordinates of the assisting cars after the garbling is done.

Table 2. Timing Results

Function	Garbling	Oblivious Transfer		Communication		Evaluation
		Garbler	Evaluator	Garbler	Evaluator	
Intersection	2.97E + 07	3.18E + 08	2.94E + 08	6.06E + 05	3.16E + 07	2.34E + 07
Range	3.65E + 05	3.06E + 08	2.83E + 08	5.40E + 04	3.55E + 05	2.96E + 05
TrilLoc	8.90E + 08	6.53E + 09	7.31E + 09	7.25E + 09	7.25E + 09	1.36E + 08

To run the BMR-based protocol, we employ the framework provided in References [14, 15]. Unlike GC, each car acts as garbler and only the lost car Q acts as the evaluator. The average number of clock cycles at different stages of the BMR protocol with the $TriLoc$ netlist is presented in Table 2. The complete protocol execution takes $8.97E + 09$ clock cycles, which translates to 2, 646ms. As expected, the BMR-based protocol have a longer run time than the GC-based protocol. Similar to the previous case, the assisting cars may wait till the end of the garbling phase before locking their coordinates. Note that in both cases the protocol execution will have to wait till all three assisting cars join. That wait time is not included in this evaluation.

Even though the evaluation is performed on a desktop PC, this protocol is practical with processors available in smart cars today. For example, Intel Atom Processor E3845, designed for in-vehicle solutions, has four cores operating at 1.91GHz and an L2 cache of 2MB [23]. The protocol requires transmission of about 1MB of data. With transmission speed in MHz range [36], the transmission time is within practical limits. The memory footprint of this operation is about 1.8MB, which can fit in the L2 cache of an Atom processor.

7 RELATED WORK

Until now, localization algorithms have been mainly used in Wireless Sensor Networks. In centroid localization, the unknown nodes location is set to the centroid of a polygon formed by the anchor nodes within a certain range. In weighted centroid localization, the centroid is calculated as the weighted mean of the coordinates of the anchor nodes [7, 43]. In triangle localization, three circles are drawn centered at three anchor nodes with the radius equal to the estimated distances from the unknown node [2, 35, 44]. The centroid of the triangle formed by the intersection is the estimated location. In this work, we employ triangle localization as it requires only three anchor nodes while for the other techniques more anchor nodes are required for accuracy.

There are a number of works that designed privacy preserving Location-Based Services based on cryptographic primitives. Methods for privacy-preserving nearest neighbor search are presented in References [12, 25]. The work in Reference [25] employs one-way Hilbert transformation to map the space of all elements to another space and resolve the query in that transformed space. It requires a trusted third party to perform the transformation in an offline phase. The method presented in Reference [12] confines each point of interest (POI) to a cell, named a Voronoi cell, such that the POI is the nearest neighbor to any point that falls within that cell. Then a regular rectangular grid is superimposed over this Voronoi diagram. A user retrieves all the Voronoi cells intersecting the region she belongs to on the grid through private information retrieval method and locally computes the nearest neighbor. Both these methods consider the privacy of the query only; the database of the POIs is assumed to be public. Three methods based on homomorphic encryption to find if two friends are nearby without revealing their locations is presented in Reference [45]. There are different trade-offs involved in these methods: they either require a semi-trusted third party or sacrifice accuracy or privacy for simplified operation.

The work in Reference [1] presents application specific solutions based on GC to several problems in location-based services. They solve basic problems like point-inclusion (whether or not one party's point is included in other party's polygon), intersection (whether or not two polygons from two users intersect), closest pair (form a pair closest to points taking one point from each set provided by two users). A GC-based method to compute the nearest neighbor of a group of people is presented in Reference [19]. In this method, two users participate in GC protocol to compute the nearest neighbor of the group. The other members of that group receive their input keys through OT from the garbler and share them with the evaluator. This creates a security threat as the collusion between only two users will reveal the location of all other members of the group. A scalable privacy preserving k-nearest neighbor search is presented in Reference [38], which utilizes a sequential description of GC [39].

8 CONCLUSION

We present the first provably secure localization method for smart vehicles. We devise two protocols that allow a lost car to compute its location with assistance from three nearby cars through SFE techniques such that locations of all the cars remain private. We employ two well-known SFE techniques named Yao's GC and BMR for the computations jointly performed by the cars to determine the location of the lost car without revealing their own locations to any other car. The two protocols show trade-off between collusion deterrence and performance. Our localization method is one of the very first location-based services that does not involve any trade-off between accuracy and privacy. We design netlists for the functions required for computation of location and compile them with conventional logic synthesis tool using custom libraries that incorporate implementations of arithmetic operations optimized for the GC/BMR. Our implementation demonstrates that the localization operation is completed within only 355ms, a time period short enough to localize moving cars.

REFERENCES

[1] Mikhail Atallah and Wenliang Du. 2001. Secure multi-party computational geometry. In *Algorithms and Data Structures*. Springer, 165–179.

[2] Paramvir Bahl and Venkata N. Padmanabhan. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, Vol. 2. IEEE, 775–784.

[3] Donald Beaver, Silvio Micali, and Phillip Rogaway. 1990. The round complexity of secure protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*. ACM, 503–513.

[4] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. 2013. Efficient garbling from a fixed-key blockcipher. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 478–492.

[5] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. 2016. Optimizing semi-honest secure multiparty computation for the internet. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 578–590.

[6] Alan Bensky. 2016. *Wireless Positioning Technologies and Applications*. Artech House.

[7] Jan Blumenthal, Ralf Grossmann, Frank Golatowski, and Dirk Timmermann. 2007. Weighted centroid localization in zigbee-based sensor networks. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing (WISP'07)*. IEEE, 1–6.

[8] Michael Brenner, Henning Perl, and Matthew Smith. 2013. hcrypt SFE project. Retrieved from https://github.com/hcrypt-project/yao.

[9] D. Brown, Geoffrey Cooper, Ian Gilvarry, Anand Rajan, Alan Tatourian, Ramnath Venugopalan, David Wheeler, and Meiyuan Zhao. 2015. Automotive security best practices. Retrieved from http://www.intel.com/content/www/us/en/automotive/automotive-security-best-practices-white-paper.html.

[10] Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. 2006. Preserving user location privacy in mobile data management infrastructures. In *Lecture Notes in Computer Science*, Vol. 4258. Springer, 393–412.

[11] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. 2002. Tools for privacy preserving distributed data mining. In *Sigkdd Explorations Newsletter*, Vol. 4. ACM, 28–34.

[12] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. 2008. Private queries in location-based services: Anonymizers are not necessary. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 121–132.

[13] Lewis Girod, Vladimir Bychkovskiy, Jeremy Elson, and Deborah Estrin. 2002. Locating tiny sensors in time and space: A case study. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*. IEEE, 214–219.

[14] Bar Ilan Cryptography Research Group. 2016. Semi-Honest-BMR. Retrieved from https://github.com/cryptobiu/Semi-Honest-BMR.

[15] Bar Ilan Cryptography Research Group. 2017. libscapi. Retrieved from https://github.com/cryptobiu/libscapi.

[16] Marco Gruteser and Dirk Grunwald. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*. ACM, 31–42.

[17] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. 2002. The anatomy of a context-aware application. In *Wireless Networks*, Vol. 8. Springer-Verlag, New York, 187–197.

[18] Y. Huang, D. Evans, and J. Katz. 2012. Private set intersection: Are garbled circuits better than custom protocols? In *Proceedings of the Network and Distributed Security Symposium (NDSS'12)*.

[19] Yan Huang and Roopa Vishwanathan. 2010. Privacy preserving group nearest neighbour queries in location-based services using cryptographic techniques. In *Proceedings of the Global Telecommunications Conference (GLOBECOM'10)*. IEEE, 1–5.

[20] Jean-Pierre Hubaux, Srdjan Capkun, and Jun Luo. 2004. The security and privacy of smart vehicles. In *Proceedings of the IEEE Symposium on Security and Privacy*, Vol. 2. IEEE, 49–55.

[21] Siam U. Hussain and Farinaz Koushanfar. 2016. Privacy preserving localization for smart automotive systems. In *Proceedings of the Design Automation Conference (DAC'16)*. ACM, 26–31.

[22] Synopsys Inc. 2015. Design Compiler. Retrieved from https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[23] Intel. 2015. Atom Processor E3845. Retrieved from ark.intel.com/products/78475.

[24] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. 2007. Preventing location-based identity inference in anonymous spatial queries. In *Transactions on Knowledge and Data Engineering*, Vol. 19. IEEE, 1719–1733.

[25] Ali Khoshgozaran and Cyrus Shahabi. 2007. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *Advances in Spatial and Temporal Databases*. Springer, 239–257.

[26] Vladimir Kolesnikov and Thomas Schneider. 2008. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming*. Springer, 486–498.

[27] Benjamin Kreuter, Abhi Shelat, Benjamin Mood, and Kevin RB Butler. 2013. PCF: A portable circuit format for scalable two-party secure computation. In *Proceedings of the USENIX Security Symposium*. USENIX, 321–336.

[28] Paweł Kułakowski, Javier Vales-Alonso, Esteban Egea-López, Wiesław Ludwin, and Joan García-Haro. 2010. Angle-of-arrival localization based on antenna arrays for wireless sensor networks. In *Computers & Electrical Engineering*, Vol. 36. Elsevier, 1181–1186.

[29] Moni Naor and Benny Pinkas. 2005. Computationally secure oblivious transfer. In *Cryptology*, Vol. 18. Springer, 1–35.

[30] Moni Naor, Benny Pinkas, and Reuban Sumner. 1999. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*. ACM, 129–139.

[31] Panagiotis Papadimitratos, Levente Buttyan, Tamás Holczer, Elmar Schoch, Julien Freudiger, Maxim Raya, Zhendong Ma, Frank Kargl, Antonio Kung, and Jean-Pierre Hubaux. 2008. Secure vehicular communication systems: Design and architecture. In *IEEE Communications Magazine*, Vol. 46. IEEE.

[32] Aanjhan Ranganathan, Nils Ole Tippenhauer, Boris Škorić, Dave Singelée, and Srdjan Čapkun. 2012. Design and implementation of a terrorist fraud resilient distance bounding system. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 415–432.

[33] Kasper Bonne Rasmussen and Srdjan Capkun. 2010. Realization of RF distance bounding. In *Proceedings of the USENIX Security Symposium*. 389–402.

[34] Peng Rong and Mihail L. Sichitiu. 2006. Angle of arrival localization for wireless sensor networks. In *Proceedings of the IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'06)*, Vol. 1. IEEE, 374–382.

[35] Ye Shang, Zhigang Liu, Jinkuan Wang, and Xianda Xiao. 2012. Triangle and centroid localization algorithm based on distance compensation. In *Proceedings of the International Conference on Information Science and Control Engineering*. IET.

[36] ITS Standards Fact Sheets. 2009. IEEE 1609—Family of Standards for Wireless Access in Vehicular Environments (WAVE). Retrieved from standards.its.dot.gov/factsheets/factsheet/80.

[37] E. M. Songhori and S. U. Hussain. 2017. TinyGarble. Retrieved from https://github.com/siamumar/TinyGarbled.

[38] Ebrahim M. Songhori, Siam U. Hussain, Ahmad-Reza Sadeghi, and Farinaz Koushanfar. 2015. Compacting privacy-preserving k-nearest neighbor search using logic synthesis. In *Proceedings of the Design Automation Conference (DAC'15)*. ACM, 36–42.

[39] Ebrahim M. Songhori, Siam U. Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. 2015. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *Proceedings of the IEEE Symposium on Security & Privacy*. IEEE, 411–428.

[40] U.S. Department of Defense. 2008. Global positioning system standard positioning service performance standard. Retrieved from https://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf.

[41] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*. IEEE, 162–167.

[42] Samee Zahur, Mike Rosulek, and David Evans. 2015. Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 220–250.

[43] Jijun Zhao, Qingwei Zhao, Zhihua Li, and Yunfei Liu. 2013. An improved weighted centroid localization algorithm based on difference of estimated distances for wireless sensor networks. In *Telecommunication Systems*, Vol. 53. Springer, 25–31.

[44] Jungang Zheng, Chengdong Wu, Hao Chu, and Peng Ji. 2010. Localization algorithm based on RSSI and distance geometry constrain for wireless sensor network. In *Proceedings of the International Conference on Electrical and Control Engineering*. IEEE, 2836–2839.

[45] Ge Zhong, Ian Goldberg, and Urs Hengartner. 2007. Louis, lester and pierre: Three protocols for location privacy. In *Proceedings of the International Workshop on Privacy Enhancing Technologies*. Springer, 62–76.

Received October 2017; revised June 2018; accepted June 2018