Don't Trust Any Car

Traffic Condition Revealing with Multiparty Computation

黃資翔
NTU CSIE
b06705057

詹洵泰
NTNU CSIE
ntnu_40747040s

曹習愛
NTU AS
b06209035

蘇鈺崴
NTU AGEC
b06607057

ABSTRACT

We propose a novel secure traffic condition revealing system to help drivers make better route decisions. The system adopts secure multiparty computation to find the median speed at each location to indicate the traffic condition. In our system, all the users' privacy is been protected in both passive and active adversaries settings. We also analyzed the time complexity and applicability for different protocols under our system setting. We revised and adapted these protocols for the system and fairness is remain for all parties. The overall protocol communication complexity for n parties in L locations with possible speed range M is $O(Ln \log M)$.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

secure multiparty computation, k^{th}-ranked element, traffic condition revealing

1 INTRODUCTION

While driving on the road, people would like to know the traffic conditions and to avoid traffic jams. In this project, we design a secure traffic condition revealing system to tackle this problem. There are several ways to exhibit the traffic condition. We can calculate the number of cars on the road, but this value disregards the different capacities of each road. We can calculate the density of cars instead. However, the system should keep the road capacity in advance which aggravates the system burden. A better method is to look at the normal speed of cars on the road. To mitigate the effect of extreme values, we find the median speed of cars on the road.

The information relies on the participation of the users. The users upload their speed at their location. By using multiparty computation, we can calculate the traffic statistics on every location. This shows the traffic conditions. The drivers can, then, find out the fastest route to get to the destination by utilizing this information.

Since the traffic condition will change at each time point, this gives a tight restriction on the time complexity of the protocol. The computation should be done efficiently and the number of communications should be minimized to correctly reflect the current traffic condition.

There are lots of security issues in this system. The user identification should not be revealed to prevent being tracked by anyone. Hence, the privacy data including the location and the speed should always be protected. We take advantage of secure multiparty computation to achieve the task. Besides, the traffic condition should not be manipulated by the attacker which may be accomplished by sending a large amount of fake data or colluding with other users.

In addition to the above problems, multiple users in the same car may cause bias in the traffic statistics. We propose a method to normalize the effect of the repeated value without identifying the duplicated cars. This avoids the user identification leak and keeps the user's privacy.

We conclude our contribution with the following

- We design a secure but practical system for traffic condition revealing.
- We analyze the communication and computation complexity for different protocols and find out the most appropriate one for our system.
- Our protocol can resist both passive and active adversaries' attacks in multiparty computation scheme.
- Our protocol retains fairness for all parties.

2 RELATED WORK

2.1 Garbled Circuit

Secure computation can be used to address the confidentiality issue. It guarantees that no party will learn more than the output. Previous works about secure computation can be traced back to Yao's work.[8] Yao's protocol, which is called Garbled circuit later, allows two parties holding private values i and j respectively to compute any polynomially computable functions $f(i, j)$. Goldreich et al. further proposed a generic protocol(GMW protocol) that allows any number of parties to achieve secure computation.[3]

However, these protocols aim at providing a generic solution to secure computation. Communication Costs on some functions may not be optimal when using these protocols. For lower Communication Costs, Specific protocols are then needed.

2.2 Secure Set Intersection

If people in the same car have the same location code, we can address the problem of multiple users in the same car by private equality testing. The equality testing is just a special case of set intersection with only one element in each set. With the use of polynomial representation of set, Kissner and Song [5] proposed an efficient multiparty secure set intersection protocol based on Paillier's homomorphic encryption [6]. The method achieves $O(n^2 \log |P|)$ for active adversary setting where P is the domain of the set element. However, we find out that the InnerCircle method can further reduce the communication complexity but without the necessity of the same location code assumption.

2.3 InnerCircle

Hallgren et al. [4] built up a decentralized protocol to detect whether a person is within a region of radius **r** around you. With the privacy-preserving property, no one needs to reveal his/her location. The technique is mainly based on homomorphic encryption to compute a distance list **L**. If a person is within a threshold radius **r**, then the decryption of **L** will contain a 0. Moreover, there is a random process that shuffles L to hind the information about “how close” a person is.

2.4 k^{th}-Ranked Element

Aggarwal et al. [1] proposed a protocol to compute the k^{th}-ranked element for multi-parties. This protocol can be done in $O(\log M)$ rounds, and with $O(n \log M)$ overhead per round, where n is the number of parties and M is the input range. In the scheme, they use secure multi-party computation to compute the summations as well as the comparisons without revealing the private data each party holds. The scheme is secure under both passive and active adversary. It can resist against active adversary by verifying whether the parties' inputs are consistent in every round. We then modify this work to fit our tasks.

To further decrease the time complexity in finding k^{th}-ranked element among parties, introducing a trusted third party to the system might be a good idea. Tueno et al. [7] show how to find k^{th}-ranked element in a star network using either garbled circuit or homomorphic encryption. The server takes a heavier communication load with $O(n \log n)$ and the client only takes $O(n)$. In our problem, the number of users is, however, much larger than the number of bits of the elements which represents the speed of each location. Hence, even though the original method has quadratic growth in the number of bits, that is $O(n(\log M)^2)$, the smaller power of the number of users is preferred.

3 PROBLEM DEFINITION

In brief, given that n parties probably moving along roads and participating in our protocols, we want our protocols to let these parties jointly compute the median velocity of L locations without leaking their private information.

3.1 Threat model

Following the definition from [2], We distinguish passive and active adversaries. A passive adversary means the adversary learns the complete information from the protocol transcript held by corrupted players, but all parties follow the protocol honestly. An active adversary means the adversary takes full control of the corrupted parties.

Both passive and active adversaries may be static or adaptive. "Static" means that the adversary chooses which parties to corrupt before the protocol execution and the set of corrupted parties should be fixed during the protocol execution. "Adaptive" means that the adversary can corrupt parties dynamically during the protocol execution. We assume that the adversary has the power to passively/actively and statically/adaptively corrupt a limited number of parties.

3.2 Security definition

To show a protocol is secure, the key idea is to show that the real world would act as good as in the ideal world. **The real world** contains n parties which are modeled as interactive Turing machines. They execute the protocol Π to compute a function f. We assume that the parties communicate under secure pairwise communication channels. Besides, all parties get a security parameter k as input. Let $\mathbf{Real}_{\Pi,\mathcal{A},k}(x_1,\ldots,x_n)$ be the output distribution induced by executing the protocol Π with adversary $\mathcal{A}$.

The ideal world contains n parties $P_1,\ldots,P_n$ and a trusted angel which is assumed to be honest. Each party sends their input to the angel. Angel then helps to compute $f(x_1,\ldots,x_n)$. Honest parties P_i simply sends x_i but corrupted parties send x_i' (where possibly $x_i' \neq x_i$) to the angel. The angel computes f and obtains $(y_1,\ldots,y_n)$. The protocol we considered are fair. That is, the adversary will receives responses only if the honest parties receive responses from the angel. Let $\mathbf{Ideal}_{f,\mathcal{A},\mathcal{S},k}(x_1,\ldots,x_n)$ be the output distribution described by the ideal world, where S is a simulator interact between the angel and the adversary.

Definition 1. *A protocol Π securely computes f if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator S in the ideal world such that for all $(x_1,\ldots,x_n)$ we have,*

$$\mathbf{Real}_{\Pi,\mathcal{A},k}(x_1,\ldots,x_n) \approx_c \mathbf{Ideal}_{f,\mathcal{A},\mathcal{S},k}(x_1,\ldots,x_n)$$

That is, for all PPT distinguisher $\mathcal{D}$, we have

$$\Big|\Pr[\mathcal{D}(\mathbf{Real}_{\Pi,\mathcal{A},k}(x_1,\ldots,x_n)) = 1] - \Pr[\mathcal{D}(\mathbf{Ideal}_{f,\mathcal{A},\mathcal{S},k}(x_1,\ldots,x_n)) = 1]\Big|$$

is negligible in k.

Note that $\mathcal{A}$ can passively/actively and statically/adaptively corrupt a limited number of parties.

3.3 Assumptions

(1) A server can at most be a passive adversary, not active.
(2) Data communication channels between server and users are authenticated and confidential.
(3) At most half of the users in each location can be corrupted.
(4) Distances between users in the same car are shorter than distances between two cars.
(5) No cars exceed the set minimum and maximum velocity in the protocol.

3.4 Other definitions

Definition 2. *(Commitment Scheme) A commitment scheme $\mathbf{Com} := (\mathbf{S}, \mathbf{R}, \mathbf{Open})$ is a protocol with two stage. In the first stage, sender S sends a commitment c of x to receiver R. In the reveal stage, sender S sends a pair of (x, d), where d is a decommitment string. Receiver R outputs $\mathbf{Open}(c, x, d) \in \{accept, reject\}$. A commitment scheme is computationally hiding if for all message x and x', commitment of x and x' are computationally indistinguishable. A commitment scheme is computationally biding if the sender cannot efficiently find two decommitment pairs (x, d) and (x', d') with $x \neq x'$ such that $\mathbf{Open}(c, x, d) = \mathbf{Open}(c, x', d') = accept$.*

Definition 3. *(Proof of Knowledge) A pair of interactive machines $\langle P, V\rangle$ is called a proof of knowledge protocol for a* **NP** *language $\mathcal{L}$ (where $R_{\mathcal{L}}$ is the witness relation), if the following holds:*

(1) ***(PPT)*** *P and V are efficiently computable.*
(2) ***(Completeness)*** $\forall k \in \mathbb{N}$, $\forall x \in \mathcal{L}$, $\Pr[\langle P, V\rangle(x, 1^k) = 1] = 1$
(3) ***(Knowledge Soundness)*** *There exists an expected polynomial-time machine **E**, called the extractor, such that there exists negligible function $negl(\cdot)$, for all efficient P^*, x, auxiliary input z, random tape r, and $k \in \mathbb{N}$, it holds that*

$$\Pr[E^{P_z^*(x,r)}(x, 1^k) \ : \ (x, w) \in R_{\mathcal{L}}] \geq$$
$$\Pr[\langle P_z^*(r), V\rangle(x, 1^k) = 1] - negl(k)$$

(4) ***(Zero-knowledge)*** *There exists an efficient simulator S and $negl(\cdot)$ such that*

$$\forall \ PPT\, V^*, \ \forall k \in \mathbb{N}, \ \forall x \in \mathcal{L}, \ S^{V^*}(x, 1^k) \approx_c \langle P, V^*\rangle(x, 1^k)$$

where $\approx_c$ represent computationally indistinguishable and $\langle P, V^*\rangle$ is the random variables of the messages that prover P sends to V and the randomness of V.

4 RESULTS

We assume that there is a central server that helps us transfer messages. Each message will be sent to the server and the server will transfer the message to the corresponding receivers. Note that by the assumption in section 3, the server can only be passive. In sections 4.1 and 4.2, we give a protocol where all parties are only passive malicious. In section 4.3, we transform the protocol from 4.1 to another protocol that allows active malicious parties. In section 4.4, we give security proof along with correctness and overhead.

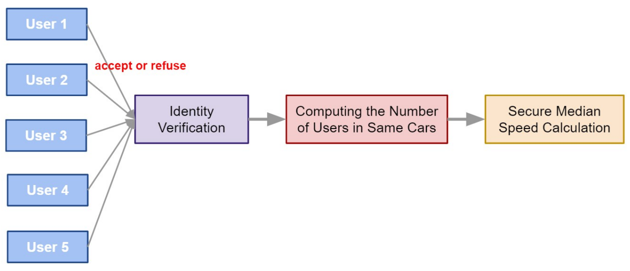

User 1
User 2
User 3
User 4
User 5
accept or refuse
Identity Verification
Computing the Number of Users in Same Cars
Secure Median Speed Calculation

Figure 1: Protocol Structure

In Figure 1, there are three stages in our protocol.

In the first stage, the central server will check the user's identity to prevent an attacker from sending numerous wrong messages to control the protocol. Furthermore, When the identity verification is done, one round of the median calculation will be proceeded by the same group of users, then the result is deterministic.

In the second stage, all participants will compute the number of different users in the same car to prevent repeated calculations. The details are presented in section 4.1.

In the third stage, we perform the secure median speed protocol and output the median speed for each location. The details are presented in sections 4.2 and 4.3.

4.1 Distance Between Two Point

In our protocol, different users in the same car or one who has multiple cellphones would cause repeated calculations and distortion for the median result. With the inspiration from Per Hallgren et al.'s work in [4], we modified the original scheme to fit our requirements.

We use Euclidean distance to calculate the distance between two users and determine whether they are at the same location. Since the location (x,y) is the private data that can't be revealed to the public, we use homomorphic encryption to hide the real (x,y) while users can still decrypt the final encrypted answer and get the distance from others.

Assume there are 2 users, A and B, where A is at $(x_A, \ y_A)$ and B is at $(x_B, \ y_B)$. The Euclidean distance **d** is defined as:

$$\mathbf{d} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$
$$\mathbf{D} = d^2 = {x_A}^2 + {x_B}^2 + {y_A}^2 + {y_B}^2 - (2x_A x_B + 2y_A y_B)$$

Then the homomorphic encryption of **D** is **E(D)**

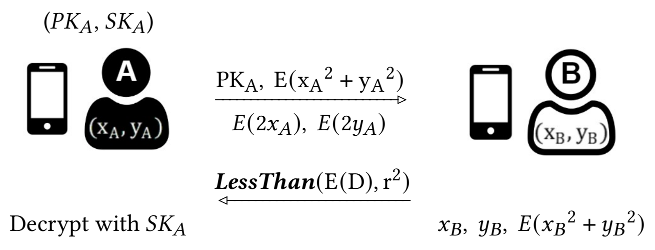
$$\mathbf{E(D)} = E({x_A}^2 + {y_A}^2) \oplus E({x_B}^2 + {y_B}^2)$$
$$\ominus \ (\ (E(2x_A) \odot \ x_B) \ \oplus \ (E(2y_A) \odot y_B) \ )$$

User A sends her public key PK_A and some encrypt data with $PK_A \ (\ E({x_A}^2 + {y_A}^2), \ E(2x_A) \ and \ E(2y_A))$ to User B. Then user B computes the **E(D)** homomorphic function with PK_A. Next, B return **E(D)** to A, then A can decrypt it with her secret key SK_A. There are still some problem in the above scheme. Since A has the secret key SK_A, then she can find out all the distances from other users. Now, we propose a function ***LessThan*** to hide all the distances from other users, and the only information remained for A is whether B is within a certain distance **r** from her.

Algorithm 1 *LessThan*$(x, \ y)$

```
$L \leftarrow E(1)$
for $i \leftarrow 0$ to $i \leftarrow y - 1$ do
    $L \leftarrow L \odot (x \ominus E(i))$
end for
$R$: a random number
$L \leftarrow L \odot E(R)$
return L
```

We set the variable **r** to a small number that the area within radius **r** represents the space of a car. In the following process, A can decrypt ***LessThan***(E(D)) with SK_A. Only if the distance is within r, the decryption is zero. Otherwise, it's a random number.


(PK_A, SK_A)
A
(x_A, y_A)
$PK_A, \ E({x_A}^2 + {y_A}^2)$
$E(2x_A), \ E(2y_A)$
B
(x_B, y_B)
LessThan$(E(D), r^2)$
Decrypt with SK_A
$x_B, \ y_B, \ E({x_B}^2 + {y_B}^2)$

Users can decrypt all the encrypted distance messages sent from others and see how many users are actually in the same car with them. Then users can compute a weighting factor $\boldsymbol{w} = \frac{1}{\boldsymbol{duplication\ number}}$, where ***duplication number*** is the number of users in the same car. The weighting factor $\boldsymbol{w}$ will be used in the next section to normalize the bias in the traffic analysis with repeated counts of cars.

Complexity in section 4.1

The communication cost is O(3n) per user, where n is the number of users. We assume all the communication between users is transmitted by the server, where the server can interact with all the users in parallel. There are three rounds of O(n) communication in the scheme. In the first round, the server will send all the other users' public key and the ***distance-requests*** (contains $E({x_A}^2+{y_A}^2)$, $E(2x_A)$ and $E(2y_A)$) to each user. In the second round, each user will use others public key to compute the ***LessThan***$(E(D), r^2)$ function as the ***distance-response*** for each user and return to the server. In the third round, the server will transfer all the ***distance-response*** to the specific users.

4.2 Multi-Party Computation Against Passive Adversary

Our secure median speed protocol is mainly based on k^{th}-ranked element scheme [1]. In this protocol, we output the median speed for each location L_j with the collaboration of all the secret data from each party P_i. In the initialization, we choose a reasonable speed range $[\alpha, \beta]$, and set the protocol current range $[a, b]$ to $[\alpha, \beta]$. Next, we compute the protocol several rounds where the updating process is very similar to binary search. In every round, we do the following steps: (1) Set the current guessing median to $m = \frac{\lceil a+b \rceil}{2}$. (2) For each party, compare its speed with the guessing median, and compute the lower (or higher) weighted number $l'_{i,j}$ (or $g'_{i,j}$). (3) Perform the secure computation to sum up $l'_{i,j}$ and $g'_{i,j}$, then check ***UNDERESTIMATE*** and ***OVERESTIMATE*** conditions. (4) Use ***UNDERESTIMATE*** and ***OVERESTIMATE*** flags to check whether the protocol is done. Otherwise, change the boundary $[a, b]$ and try a new round with the new guessing median.

We sketch our protocol in the following.

Protocol 1 Secure Median Speed

```
Inputs
    Speed range $[\alpha, \beta]$ is public, and each party $P_i$ would keep
    its own private (location, speed) data pair.
Output
    Median speed of each location $L_j$.
Initialization
  (1) Determine a reasonable speed range $[\alpha, \beta]$.
  (2) Set location $L_j$'s current range $[a_j, b_j]$ to $[\alpha, \beta]$.
Loop (repeat until Done)
  (1) Guess the median $m_j = \frac{\lceil a_j+b_j \rceil}{2}$.
  (2) Each party $i$ compute its secrets for location $L_j$:
          $(l'_{i,j}, g'_{i,j}, w_{i,j})$
      where $l'_{i,j} = w_{i,j} \cdot l_i$ and $g'_{i,j} = w_{i,j} \cdot g_i$
      $l_i$: 1 if the speed is strictly lower than $m_j$, otherwise 0.
      $g_i$: 1 if the speed is strictly larger than $m_j$, otherwise 0.
      $w_{i,j} = \begin{cases} 0 & \text{if the party is not at location } L_j \\ 1 & \text{if the party is at location } L_j \text{ and no duplicates} \\ \frac{1}{duplication\ number} & \text{otherwise} \end{cases}$
  (3) Securely computes 2 boolean bits with GMW circuit for each
      location.
          $\textbf{\textit{UNDERESTIMATE}} := \sum_i g'_{i,j} \geq k_j + 1$
          $\textbf{\textit{OVERESTIMATE}} := \sum_i l'_{i,j} \geq k_j$
      where $k_j = \frac{1}{2} \sum_i w_{i,j}$
  (4) Evaluation
    (a) If both UNDERESTIMATE and OVERESTIMATE are
        false, then compute Done to exit the loop, and output the
        final median $m_j$.
    (b) If OVERESTIMATE is true, then update current speed
        range to $[a_j, m_j - 1]$
    (c) If UNDERESTIMATE is true, then update current speed
        range to $[m_j + 1, b_j]$
```

In **protocol 1** loop step (3), all users securely use the GMW circuit to compute the two bits information, ***UNDERESTIMATE*** and ***OVERESTIMATE***, without disclosing any user's private data. With this secure multi-party computation scheme, $l'_{i,j}$, $g'_{i,j}$ and $w_{i,j}$ will always remain secret, and users' location and speed will be kept private.

4.3 Multi-Party Computation Against Active Adversary

The idea is that we compel each party to fix its private secret and randomness. When the randomness is fixed, the behaviors of the parties become deterministic. Note that the randomness will only be used at step (3) in protocol 1. We introduce a new protocol called the "Coin-flipping Phase" that should run in the very beginning. Let **Com** be a commitment scheme that is computationally hiding and computationally binding.

Protocol 2 Coin-Flipping Phase

Inputs

Each party P_i decides its own private x_i := (location, speed) data pair.

Output

Each party P_i obtains a randomness string r_i that should be used in the protocol afterward.

For each party ***Do:***

(1) Each party P_i sample $r_{P_i} \leftarrow \{0,1\}^m$, where m is the security parameter.
(2) Each party P_i and the server interact in **Com**, so that the server learns c_i, the commitment of (x_i, r_{P_i}). Besides, P_i proves to the server that he/she knows the opening for c_i. If the proof is not accepted, the server aborts.
(3) For each party, the server sample $r_{S_i} \leftarrow \{0,1\}^m$ and sends r_{S_i} to party P_i.
(4) Each party P_i compute $r_i := r_{P_i} \oplus r_{S_i}$.

After running the coin-flipping phase (protocol 2), we will run the protocol 1. However, in addition, when the party sends messages to the server, the party has to additionally give a "proof" which means he/she follows the protocol honestly. Let $\langle P_{\text{knowledge}}, V_{\text{knowledge}} \rangle$ be a proof of knowledge protocol that is knowledge soundness and computationally zero knowledge.

Protocol 3 Proof of honesty

For each party P_i and l-th message m sent by P_i ***Do:***

(1) Let τ be the messages exchanged so far between the party and the server. Let f_l be a function such that the honest party will sends $f_l(x_i, \tau; r)$ to the server.
(2) Define $L_l := \{(c, \tau, r, m) \mid \exists\, x, r_P, d \ s.t.\ \textbf{Open}(c, (x, r_P), d) \ \wedge\ m = f_l(x, \tau; r_P \oplus r)\}$. The party P_i proof to the server that $(c_i, \tau, r_{S_i}, m) \in L_l$ using the proof of knowledge $\langle P_{\text{knowledge}}, V_{\text{knowledge}} \rangle$. Denote the transcript of this proof as $\theta_{i,l}$. The server aborts if $\theta_{i,l}$ is not accepting.

Now, we combine the three protocols against an active adversary.

Protocol 4 Secure Median Speed (Against Active Adversary)

Inputs

Speed range $[\alpha, \beta]$ is public, and each party P_i would keep its own private (location, speed) data pair.

Output

Median speed of each location L_j.

Do

(1) Run the coin-flipping phase (protocol 2). The server learns $c_1, \ldots, c_n$ and each party P_i learns r_{S_i}.
(2) If the server aborts for some proof, the server forbids the party to participate anymore.
(3) Run the honest secure median speed protocol (protocol 1). Besides, when the party sends a message m to the server, the party proves that he/she is honest by running protocol 3.
(4) If the server aborts for some proof $\theta_{i,l}$, the server forbids the party to participate anymore. The remaining parties go back to step (3) and run again.

Note that step (3) may run many times. However, when a party gives wrong proof, he/she will be detected immediately by the server. The server will forbid the party to participate in the protocol anymore. That is, step (3) will be re-run at most the maximum number of users during the lifelong time.

To speed up the running time when the server detects malicious behaviors, we can continue running step (3) instead of re-running it. This may make some of the locations where the malicious parties claim to locate unable to obtain answers. We can, however, use the previous speed to approximate. If we run the protocol once every short period, the speed should not change significantly. Also, by this fact, there is another method to speed up the running time. That is setting the initial guess of median m to m_{prev}, where m_{prev} is the median of the previous round (in protocol 1, loop step (1)).

4.4 Analysis of Protocol 1 and 4

Correctness: The correctness is followed by the following:

(1) Correctness of protocol 3 in [1]
(2) $\sum_{i=1}^{n} w_{i,j}$ is the number of cars in position j
(3) $l'_{i,j} \in \{0, w_{i,j}\}$, $g'_{i,j} \in \{0, w_{i,j}\}$
(4) $l'_{i,j}$ and $g'_{i,j}$ cannot simultaneously equal $w_{i,j}$.

Overhead: The number of rounds is $O(log\,(\beta - \alpha))$ in asymptotic analysis since any malicious party has only one chance of not following the protocol. The message complexity is $O(Ln)$, which is linear to the number of locations and users.

Theorem 4.1. *Protocol 1 securely computes the median against a passive adversary with at most $n-1$ malicious parties.*

Proof. It is sufficient to construct an efficient simulator S such that given the oracle access to the adversary, S can generate a transcript indistinguishable from the honest protocol. First, consider the messages for computing $w_{i,j}$. We assume that the public key encryption scheme is secure (e.g. IND-CPA secure). Besides, the duplication number of each malicious party is given as an input to the simulator, say *duplicate*. This may be a minor leakage but should not break the security in general (The adversary could know this information in the real world by counting how many people are in the same car. However, if a user has many cellphones and participates as multiple parties, this information may leak to others). The simulator interacts with the adversary as follows: First, the simulator chooses arbitrary private data (x_A, y_A) and sends $E(x_A^2 + y_A^2), E(2x_A), E(2y_A)$ to the adversary. Since the encryption scheme is secure, this message is indistinguishable from honest messages. At the same time, the simulator receives the adversary's first messages. Second, the simulator sends *duplicate* many encryptions of 0 and others are encryption of non-zero random numbers. This is also indistinguishable from the honest messages. Next, the simulator wants to simulate the secure median speed protocol. It can be simulated successfully since we use secure sub-protocols (GMW circuits) and the fact of composition theorem (which is also used in theorem 4 in [1]). □

Theorem 4.2. *Protocol 4 securely computes the median against an active adversary with at most $n-1$ malicious parties.*

Proof. This is indeed true by the following:

(1) By theorem 4.1
(2) Since the proof of knowledge protocol is knowledge soundness, the probability that an adversary gives an acceptable proof for an instance $(c, \tau, r, m) \notin L_l$ is negligible.
(3) Since the proof of knowledge protocol is knowledge soundness, the simulator with a knowledge extractor could extract the adversary's randomness for computing f_l.
(4) There are only polynomial many proofs.

□

5 CONCLUSION AND FUTURE WORK

It is worth highlighting that our protocol is secure against $n-1$ malicious parties. Besides, our protocol is fair (i.e. adversary gets output only if honest parties get output) but the protocol in [1] is not fair. The fair protocol is simply followed by the fact that the central server cannot be active. The main drawback of our protocol and also the intrinsic of multi-party computation is that the adversary may lie to their private data but honestly follow the protocol. This, unfortunately, cannot be detected by any multi-party computation protocol. Hence, we have to assume that there are at least half of honest parties in each area.

Some future works remain to be solved:

(1) The time complexity of our protocol is linear to the number of locations. If we can compute many but the same secure functions more efficiently, then the time complexity can be improved.
(2) The burden of overhead is mainly on the prover. Is it possible that the server could compute for us? A possible approach is to use homomorphic encryption with multi-key.

Both of the possible directions above seem to be hard since there is barely any research.

REFERENCES

[1] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. 2004. Secure Computation of the k^{th}-Ranked Element. In *Advances in Cryptology - EUROCRYPT 2004 (Lecture Notes in Computer Science, Vol. 3027)*, Christian Cachin and Jan L. Camenisch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 40–55.

[2] Ronald Cramer and Ivan Damgård. 2005. Multiparty computation, an introduction. In *Contemporary cryptology*. Springer, 41–87.

[3] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing* (New York, New York, USA) *(STOC '87)*. Association for Computing Machinery, New York, NY, USA, 218–229. https://doi.org/10.1145/28395.28420

[4] Per Hallgren, Martín Ochoa, and Andrei Sabelfeld. 2015. InnerCircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*. 1–6. https://doi.org/10.1109/PST.2015.7232947

[5] Lea Kissner and Dawn Song. 2005. Privacy-Preserving Set Operations. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology* (Santa Barbara, California) *(CRYPTO'05)*. Springer-Verlag, Berlin, Heidelberg, 241–257. https://doi.org/10.1007/11535218_15

[6] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology — EUROCRYPT '99*, Jacques Stern (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 223–238.

[7] Anselme Tueno, Florian Kerschbaum, Stefan Katzenbeisser, Yordan Boev, and Mubashir Qureshi. 2020. Secure Computation of the k^{th}-Ranked Element in a Star Network. In *Financial Cryptography and Data Security*, Joseph Bonneau and Nadia Heninger (Eds.). Springer International Publishing, Cham, 386–403.

[8] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 162–167. https://doi.org/10.1109/SFCS.1986.25

' ' ”