On Privacy Preserving Convex Hull

Sandeep Hans, Sarat C. Addepalli, Anuj Gupta and Kannan Srinathan

Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad, India
{sandeep, sarat_a, anujgupta}@research., srinathan@iiit.ac.in

Abstract

Computing convex hull for a given set of points is one of the most explored problems in the area of computational geometry (CG). If the set of points is distributed among a set of parties who jointly wish to compute the convex hull, each party can send his points to every other party, and can then locally compute the hull using any of the existing algorithms in CG. However, such an approach does not work if the parties wish to compute the convex hull securely, i.e., no party wishes to reveal any of his input points to any other party apart from those that are part of the answer. The problem of secure computation of convex hull for two parties was first introduced by Du and Atallah (NSPW '01). The first solution to the problem was given by Wang et. al (ARES '08). However, the proposed solution was based on well known algorithms for computing convex hull in CG which are proven to be sub-optimal. We propose a new solution for secure computation of convex hull with a considerable improvement in computational complexity. We further show how to extend our two-party protocol for the case of any number of parties.

Keywords. Secure multiparty computation, Privacy preserving computational geometry, Convex hull.

1. Introduction

With the growth of the Internet, need for cooperative computing has increased considerably. The task of jointly computing some function on the inputs of the participating players is trivial, as all the players can send their inputs to a designated player, who computes the required function on these inputs and gives back the output to each player. However, what if the players are mutually distrusting? Specifically, what if the players are not willing to reveal their inputs but still wish to jointly compute some function? The area of Secure Multi Party Computation (SMPC), introduced by Yao [1], essentially aims to address such questions. Informally, the problem is: a set of $n$ players wish to jointly evaluate some function correctly on their local inputs, such that at the end of the evaluation, no party gains any additional information about the local input of any other player apart from what is implied from the output. It is assumed that not all players try and gain additional information. Traditionally, such players are modeled via a fictitious entity called the adversary, that may control a subset of players. An adversary that controls up to any $t$ of the $n$ players is denoted by $t$-adversary. A player is said to be non-faulty if and only if he faithfully executes the protocol delegated to him and does no more. There exists a very rich literature on SMPC. As the first solution to the problem, Goldreich et. al [2] presented a protocol that allows $n$ players to securely compute an arbitrary function against a computationally bounded, active $t$-adversary if and only if $t < n/2$. The problem has since been studied under various settings such as [2], [3], [4], [5], [6], [7], [8] giving both possibility (protocols) and impossibility results.

However, a large part of the work has focused on solving any function. As noted by Goldreich [2] too, these solutions are many a times practically infeasible. Motivated from this, work was initiated to find feasible solutions to specific problems. Some examples are Privacy Preserving Data Mining [9], Privacy Preserving Graph Algorithms in the Semi-honest Model [10], Privacy Preserving Cooperative Scientific Computation [11], Privacy Preserving Set Union [12], Privacy Preserving Electronic Surveillance [13], Privacy Preserving Cooperative Statistical Analysis [14] to name a few. One of many such areas is Privacy Preserving Computational Geometry (PPCG). As the name suggests, it aims to solve problems of Computational Geometry in a secure manner. Some of work done in this area being [15], [16], [17].

One of the most fundamental problems of computational geometry is to compute the convex
hull for a given a set of points. The problem was first solved efficiently by Chand and Kapur [18]. Later on, the solution was improved in many subsequent papers [19], [20], [21]. Consider the following problem: Given two players Alice and Bob, where Alice has \( m \) points in a plane and Bob has another \( n \) points in the same plane. Both want to jointly compute the convex hull for the \( m + n \) points; however neither of them wants to disclose any more information about their point sets to the other party than what could be derived from the result. This problem was first introduced by [16]. Wang et. al [22] gave two solutions for finding the convex hull securely between two players. Their protocols take \( O((m + n)hD) \) and \( O(D(m^2 + n^2)) \) time, where \( m \) and \( n \) are the sizes of \( A \)'s and \( B \)'s point sets respectively and \( D \) is the computational complexity of Yao’s protocol [1].

**Organization of the paper:** In section 2 we cover some preliminaries and definitions. In this section, we formally introduce the model in which we work followed by the problem statement. Section 3 explains the protocol, followed by section 4, where proof of correctness and privacy of the same are given. Complexity analysis of the protocol is done in section 5. We extend our protocol to the multiparty setting in section 6.

2. Preliminaries

**Notion of Security:** Given \( n \) players, each holding an input value \( x_i \), they want to calculate a function \( f(x_1, \ldots, x_n) \), without revealing their inputs. Ideally, this can be achieved using an incorruptible third party whom everyone trusts. This player is known in literature as a *Trusted Third Party (TTP)*. In an ideal setting, all the players send their inputs to the TTP, the TTP then calculates the function, and sends the output to all the players. Unfortunately, no such player exists in reality. The field of Secure Multiparty Computation gives protocols, which can simulate the actions of the TTP, among the players in the real world. A protocol is said to be secure if for every player, the information that a player gets in the real world is no more than the information that it would have got in the ideal world.

**Convex Hull:** The convex hull of a finite point set \( S \) is the smallest convex polygon \( W \) that contains \( S \), as shown in figure 1. The convex hull of a set of points \( S \) in \( n \) dimensions is the intersection of all convex sets containing \( S \). For \( N \) points \( p_1, \ldots, p_N \) the convex hull \( C \) is then given by the expression

\[
C = \left\{ \sum_{j=1}^{N} \lambda_j p_j : \lambda \geq 0 \forall j \text{ and } \sum_{j=1}^{N} \lambda_j = 1 \right\}
\]

**Our Model:** We start with two players \( A \) and \( B \), with private inputs as the set of points \( P_A = \{a_1, a_2, \ldots, a_m\} \) and \( P_B = \{b_1, b_2, \ldots, b_n\} \) respectively. Each of the two players can be modelled as an Interactive Turing Machine (ITM), connected by a completely reliable and secure communication channel. Communication over the network is assumed to be synchronous. That is, the protocol is executed in a sequence of rounds where in each round, a player can perform some local computation, send new messages to all the players, receive messages sent to him by players in the same round, (and if necessary perform some more local computation), in that order. In the send phase of each round, players write messages onto their outgoing communication tapes, and in the receive phase, players read the content of their incoming communication tapes. The players are assumed to be passively corrupt, that is, honest but curious. They follow the designated protocol diligently, but are curious to know the input of the other party. Both the players are assumed to be computationally bounded.

**Secure Two Party Convex Hull:** Given 2 players, each having a set of points on the \( x-y \) plane, a protocol is said to solve the secure two party convex hull problem if the following two condition holds:

- **Correctness:** Both the players get all the points on the vertices of Convex Hull.
- **Privacy:** None of the players gets any information on other players’ points, other than the points on the Convex Hull.

3. Privacy Preserving Convex Hull Protocol

Player \( A \) has \( m \) points and player \( B \) has \( n \) points. They want to calculate the convex hull of all the \( (m + n) \) points without revealing their respective
points, other than those points which will be on the convex hull.
Our protocol is based on Timothy Chan’s[21] algorithm for finding the convex hull. All the points on the joint convex hull will include only those points that are on individual convex hulls.

3.1. Conventions Used

Player A has m points and player B has n points. Let out of those m and n points, \( m_1 \) and \( n_1 \) lie on their individual convex hulls. These points will be numbered from \( a_1 \) to \( a_h \) and \( b_1 \) to \( b_i \) in clockwise direction, starting from the topmost point in the convex hull, i.e., \( a_1(b_1) \) is the point on \( P_A(P_B) \) with maximum y-coordinate respectively. And if there are more than one points with maximum y-coordinate, then \( A(B) \) takes \( a_1(b_1) \) as the rightmost(with maximum x-coordinate) of those.

- \( X \) and \( Y \) are player variables. These point to player \( A \) or \( B \).
- \( P_A \) and \( P_B \) refer to convex hulls generated on points of players \( A \) and \( B \) respectively. Thus, \( P_X \) and \( P_Y \) refer to \( P_A \) and \( P_B \) if \( X = A \) and \( Y = B \) and vice-versa.
- \( ycord(p) \) is the y-coordinate of point \( p \). Similarly \( xcord(p) \) refers to the x-coordinate of point \( p \).
- \( \theta_{ab} \) is the angle of inclination of the line \( a\bar{b} \) w.r.t. positive x-axis.
- \( x_c \) and starting.point are point variables referring to points in the plane.
- Each step in the protocol will be either a local step or co-operative step. The first line of each step is written accordingly.
- \( d(a,b) \) denotes the euclidean distance between two points \( a \) and \( b \).
- COMPARE\((a,b)\) denotes comparison of \( a \) and \( b \) using Yao’s protocol.
- SWAP\((X,Y)\) denotes exchange of roles between players \( X \) and \( Y \).

3.2. The Protocol

The protocol starts with both players generating their individual convex hulls. In each iteration they start with a point which is on the final hull and calculate next point on the hull. The variable \( x_c \) stores the current point of hull into consideration. The next point will either be on the same player’s individual hull or on the other player’s hull where a tangent can be drawn from \( x_c \). If we move in clockwise direction, the next point will be the point which makes largest angle on x-axis with \( x_c \). This is ensured by comparing the cosines of the respective angles, as the angles with the positive x-axis always lie between \( 0^\circ \) and \( 180^\circ \), and the cosine function is monotonically decreasing in this range. The formal protocol is given in Algorithm 1.

- **Step 1** : Both the players \( A \) and \( B \) locally compute the convex hulls of their respective point sets, \( P_A \) and \( P_B \). This can be done using Chan’s algorithm [21].
- **Step 2** : Players \( A \) and \( B \) compare their topmost points using Yao’s protocol, and decide upon the starting point. This point has the highest y-coordinate of all the \( P_A \cup P_B \) points, and hence will be on the joint convex hull.
- **Step 3** : The current point is published, since it is known to be on the convex hull.
- **Step 4** : If \( x_c \) is a part of the player’s private input set, he selects the next (in the clockwise direction) point on the individual convex hull. Otherwise, he computes the tangents from the point \( x_c \) to the individual convex hull. This can be done, since \( x_c \) is a point on the joint convex hull, and hence cannot lie inside the individual convex hull. Also, the computation of tangents can be done in \( O(\log |P_Y|) \) time. If both the tangential points lie above or below \( x_c \), the tangent with the greater angle of inclination is selected. Else, the tangent with the lesser angle of inclination is selected. If more than two points of \( P_Y \) lie on the tangent, the farther one is selected.
- **Step 5** : Both players calculate the cosines of the angles of inclination of the line segments selected in Step 4.
- **Step 6** : Both players compare the cosines calculated in Step 5, through Yao’s protocol, and decide on the next point of the hull, depending on their respective orientations.

The steps 3 to 6 are repeated until the starting point is reached.

4. Proof

4.1. Correctness

All the points on the joint convex hull are on the individual convex hulls too. Thus, no point is missed. The top-most point of all the \( m+n \) points will always be on the hull. This justifies the selection of the highest point in Step 2. The next point on the joint hull (in clockwise direction) will surely be on one of the individual hulls. If it is on the same individual hull as
Secure Two Party Convex Hull

Algorithm 1: Secure Two Party Convex Hull

1. LOCAL
Both $A$ and $B$ calculate their respective individual convex hulls - $\mathbb{F}_A$ and $\mathbb{F}_B$.

2. CO-OPERATIVE
$A$ and $B$ compare $y_{cord}(a_1)$ and $y_{cord}(b_1)$ using Yao’s protocol.
if $y_{cord}(a_1) > y_{cord}(b_1)$ then
    $x_c = a_1; X = A; Y = B$;
else
    $x_c = b_1; X = B; Y = A$;
end if
starting point = $x_c$;
repeat
3. CO-OPERATIVE
$X$ sends $x_c$ to $Y$.
4. LOCAL
If $x_c$ lies on $Y$, then let $y_i$ be the point on $Y$’s polygon next to $x_c$ in clockwise direction.
Otherwise, $Y$ generates tangents from $x_c$ on $Y$’s polygon. If $x_c$ is collinear to one of the edge $y_jy_{j+1}$, then it considers $y_{j+1}$ as the tangential point. $Y$ chooses the tangent with more angle of inclination, if both the tangential points are above or below $x_c$ and it chooses the tangent with less angle of inclination if one of the tangential point is above and the other below $x_c$. Let the point on $\mathbb{F}_Y$ where that tangent touches be $y_i$.
5. LOCAL
Both $X$ and $Y$ calculate $\cos(\theta_{x_c,x_{c+1}})$ and $\cos(\theta_{x_c,y_i})$.
6. CO-OPERATIVE
COMPARE($\cos(\theta_{x_c,x_{c+1}})$, $\cos(\theta_{x_c,y_i})$)
if $\cos(\theta_{x_c,x_{c+1}}) < \cos(\theta_{x_c,y_i})$ then
    $x_c = x_{c+1}$;
else if $\cos(\theta_{x_c,x_{c+1}}) > \cos(\theta_{x_c,y_i})$ then
    $x_c = y_i$;
    SWAP ($X$, $Y$)
else if $\cos(\theta_{x_c,x_{c+1}}) = \cos(\theta_{x_c,y_i})$ then
    COMPARE($d(x_c, x_{c+1})$, $d(x_c, y_i)$)
if $d(x_c, x_{c+1}) > d(x_c, y_i)$ then
    $x_c = x_{c+1}$;
else
    $x_c = y_i$;
    SWAP $X$ and $Y$;
end if
end if
until $x_c \neq$ starting point.

that of $x_c$, then it will be the next adjacent point on the hull. If it is on the other individual hull, then it will be a tangent from current point to the other hull. This is exactly how we are calculating the next point on the final hull. The starting point(starting point) will eventually be considered again after $x_c$ traverses all the other points on the joint convex hull. This ensures that the protocol will terminate at that point. This completes the correctness part.

4.2. Privacy
The interaction between the two players happens in one of two cases: either when one sends a point on the convex hull to the other player, or, comparison of two points or cosine of angles using the Yao’s protocol. Notice that, the points that lie on the convex hull will eventually have to be known to both the players. This implies that there is no breach of privacy when exchanging the points on the convex hull. The other interaction between the two parties is secure, since Yao’s protocol is secure.

5. Complexity Analysis
Timothy Chan [21] gave an optimal algorithm for finding the Convex Hull of a set of points, and thus also established the lower bound for the output sensitive version of the problem. Our protocol is based on his approach.
Step 1 takes $O(m \log mh)$ and $O(n \log nh)$ steps for $A$ and $B$. Step 2 and 7 take $O(Y)$ time, the computational complexity of Yao’s protocol. Step 4 takes $O(\log m_h)$ or $O(\log n_h)$, depending on whose input $x_c$ is a part of. Rest of the steps take constant time. A point on the final hull is added in each iteration. Thus, total number of iterations is $h$. The total complexity comes out to be $O(m \log mh + n \log nh + h(Y + \log mh + \log nh))$. Wang [22] gave two solutions for finding the convex hull securely between two players. However, their protocols take $O((m + n)hY)$ and $O(Y(m^2 + n^2))$ time. If we take $m \sim n \sim N$ and $mh \sim nh \sim h$, the complexity becomes $O(N \log (h) + hY + h \log (h))$ which is clearly much better than their $O(NhY)$ and $O(YN^2)$. Also, in the worst case (where $h \sim N$), these go up to $O(YN^2)$, whereas our approach takes $O(N \log N + NY)$ time, which is a significant improvement.

6. Extension to Multi-Party Setting
We now address the problem of securely computing convex hull among a set of $n$ players. Specifically we
show to use the two party protocol for the $n$ party setting. Here each player starts with some points. At the end of the protocol they wish to compute the Convex Hull of the union of all the points such that no player can learn anything about input of any other player apart from what can be deduced from the output. At first sight one may feel that the protocol given above for the two party case can be used as it is i.e. first run this protocol for every two players, then a pair of player run it with every other pair and so on. Not only such an approach will be far from optimality but will be insecure too. Each player will be able to find the orientation of all other players’ points with respect to his own points. However, this is not allowed here since the only information one player should get is the set of points on the joint convex hull and nothing more.

Let there exist a protocol $\text{find\_max}$, which on giving $n$ values as input, reports the maximum in secure manner i.e. each player at the end of running $\text{find\_max}$ get only the maximum value among the $n$ values and nothing else. In our present two-party protocol, replace all the comparisons that are done using Yao’s protocol with $\text{find\_max}$ protocol. Our claim is that the modified protocol correctly and securely computes convex hull for $n$ players. Correctness of the modified protocol stems from the fact that only change that has been made is to replace Yao’s protocol with $\text{find\_max}$ protocol. Given $\text{find\_max}$ computes maximum value correctly, modified protocol solves convex hull. Privacy also follows from similar argument.

7. Conclusion

In this paper, we present a privacy-preserving protocol for the secure computation of convex hull. We have shown that our algorithm reaches better computational lower bounds than extant literature. As this protocol is based on an output sensitive optimal convex hull algorithm [21], we conjecture that this protocol is close to optimal. The formal proof of lower bounds for the secure computation of convex hulls is left as an open problem.

References


