Privacy-Preserving Collision Detection of Two Circles*

XU Wei-jiang
Depart.of Comp. Sci. & Tech., USTC
NHPCC 416, East Campus
USTC, Hefei, 230026, PRC
86-551-3602445
wjxu@mail.ustc.edu.cn

HUANG Liu-sheng
Depart.of Comp. Sci. & Tech., USTC
NHPCC 416, East Campus
USTC, Hefei, 230026, PRC
86-551-3602445
lshuang@ustc.edu.cn

JING Wei-wei
Depart.of Comp. Sci. & Tech., USTC
NHPCC 416, East Campus
USTC, Hefei, 230026, PRC
86-551-3602445
wwjing@mail.ustc.edu.cn

YAO Yi-fei
Depart.of Comp. Sci. & Tech., USTC
NHPCC 416, East Campus
USTC, Hefei, 230026, PRC
86-551-3602445
yannyyao@mail.ustc.edu.cn

ABSTRACT
The proliferation of the network has opened up great opportunities for cooperative computation. But privacy concerns often prevent different parties from sharing their data in order to do cooperative computation tasks. Secure multi-party computation deals with the privacy concern in cooperative computation while ensuring correctness of the computation and that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output [10]. This paper addresses the problem of privacy preserving collision detection of two circles for the first time, which is an important problem in privacy preserving computational geometry. Four protocols are presented in this paper to solve the problem, and their correctness and security are also analyzed. The experimental results illustrate that our method for detecting collision of two moving circles is very effective.

Categories and Subject Descriptors
F.m [**Theory of Computation**]: Miscellaneous; E.3 [**Data**]: Data Encryption; K.4 [**Computers and Society**]: Public Policy Issues

General Terms
Algorithms, Security

Keywords
Privacy-Preserving, Collision Detection, Secure Multi-party Computation

1. INTRODUCTION
Cooperative computation, where people are cooperating with each other to conduct computation tasks based on the inputs they each supply, is one of the most important fields in computer science. In recent years, the growth of the Internet has opened up great opportunities for cooperative computation. But these computation tasks could occur between partially trusted parties, or even between competitors. All the participants only want to get the final result, but don't wish to reveal any privacy information. The problem is trivial if there is a trusted third party to conduct the computation. However, there is not such a party in most contexts, and privacy concern will become a primary problem.

Secure multi-party computation techniques aim at the problem, and can provide useful solutions. Generally speaking, secure multi-party computation (SMC) deals with the privacy concern in distributed environment while ensuring correctness of the computation and that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output [10]. SMC was first proposed by A. C. Yao in [1], and has been widely studied in theory. The theoretic general solution (circuit evaluation) can apply to every SMC problem; however, due to its high complexity problem-specific solutions should be developed [9].

* Supported by the National Natural Science Foundation (No. 60573171), the Science Foundation of Anhui Province (No. 070412043), the Ph.D. Program Foundation of Ministry of Education of China (No. 20060358014), and the China Postdoctoral Science Foundation (No. 20060390700).

Privacy-Preserving Computational Geometry (PPCG) which considers how to solve computational geometry problems in a cooperative environment while preserving every participant's private data is a class of specific SMC problems. The entities in computational geometry (points, lines, circles, polygons, etc.) are abstract representations of real entities. It is well known that, computational geometry has become an integrated discipline, and plays an important role in many applications such as computer graphics, CAD/CAM, machine learning, and VLSI. Computational geometry can solve many practical problems, and has become a new methodology. Collision detection is an important problem in computational geometry, and is widely used in many applications including intelligent mobile robots, computer cartoon technology, and computer games.

In this paper, we concentrate on the collision detection of two moving circles. The problem of privacy preserving collision detection of two circles can be defined as follows: Suppose two participants are Alice and Bob. Alice has a circle *P*, and Bob has a circle *Q*. They both want to determine whether *P* and *Q* will collide while moving, but neither of them is allowed to learn any additional information about the other participant's circle (including its moving path and velocities during moving).

We first outline related work and simply introduce basic SMC protocols used in this paper. In section 3, we formally give the overall solution and also analyze the correctness and the security of the protocols. The experimental results are given in section 4. We conclude by summarizing the contributions of this paper as well as suggestions for future research.

2. RELATED WORK

The protocols listed below serve as important building blocks in our solution.

2.1 Secure Scalar Product Protocol

Scalar product protocol was first proposed in [8], and has been widely used in SMC in recent years. It can be described as follows: Alice has a vector $X = (x_1, x_2, \cdots, x_n)$ and Bob has a vector $Y = (y_1, y_2, \cdots, y_n)$. They both want to compute and obtain the scalar product $X \cdot Y = \sum_{i=1}^{n} x_i y_i$, while the private vector of each participant can never be obtained by the other participant.

According to the share mode of the result, there are two versions of scalar product protocol: 1) both participants know the scalar product; 2) the result is shared by them, that is, Alice gets $u = X \cdot Y + v = \sum_{i=1}^{n} x_i y_i + v$ where v is random and known to Bob only while the scalar product is unknown for both of them. The latter form is usually used in order to do further computation when secure scalar product protocol is a sub-protocol in an algorithm (because in this form there is no disclosed information even from the result).

Scalar product protocol has received great attention, and has been discussed and implemented in [2, 4, 5, 7, 8]. Due to its diversity of implementation, we assume its time complexity and communication complexity are T_{scalar} and C_{scalar} respectively.

2.2 Millionaires' Protocol (Secure Comparison Protocol)

Consider two participants having their own privacy numerical values. The problem of secure comparison is to securely compare the two values without revealing any additional information about the private values to the other participant. In [1], A. C. Yao firstly presented this problem and described it as *millionaires' problem*, that is, two millionaires want to know who is richer, but don't want to disclose any additional information about each other's wealth. Yao's solution needs exponential time, and several constant-round solutions have been proposed in resent years [3, 6]. We assume its time complexity and communication complexity are $T_{comparison}$ and $C_{comparison}$ respectively.

2.3 $(Z+V)$-Disguise

$(Z+V)$-disguise scheme [11] consists of a series of secure two-party protocols. Suppose two participants are Alice and Bob. In this scheme, all intermediate results are shared by the two participants. That is, for a intermediate result Z_i, Alice only knows $Z_i + V$ while Bob only knows *V*. In that way, the intermediate results don't disclose any additional information, which is very important in complex computations. The crucial protocol is $U \cdot Z$ protocol which can be described as follows:

Input: Alice knows $Z_1 + V_1$ and Bob knows V_1 and *U*.

Output: Alice gets $U \cdot Z_1 + V$ and Bob gets *V*, where *V* is a data item consisting of random numbers, and Alice don't know *V*.

This protocol is used as a building block in $(Z+V)$-disguise scheme. Because Alice does not know *U*, the computation can still disguise the secret Z_1 even if *V* is zero. In fact, when we use the computation, we will let *V* be zero sometimes. We assume the time complexity and communication complexity are T_{UZ} and C_{UZ} respectively.

We also use the Z^{-1} protocol in the following section, which can be described as follows:

Input: Alice knows $Z_1 + V$ and Bob knows V.

Output: Alice gets *X* and Bob gets *Y*, satisfying $X = Z_1^{-1} + Y$, but neither of them knows Z_1^{-1}.

Z^{-1} protocol is implemented with two calls of $(U \cdot Z)$ -protocol, so the time complexity and communication complexity are $2T_{UZ}$ and $2C_{UZ}$ respectively.

3. PRIVACY PRESERVING COLLISION DETECTION OF TWO CIRCLES

3.1 The Framework of Solution

Suppose the two participants are Alice and Bob. Alice has a circle *P* with radius R_p, and Bob has a circle *Q* with radius R_q. For convenience, we suppose *P* is outside of *Q* and *Q* is outside of *P* (The idea of this paper can also apply to the situation where one circle contains the other one). They both want to determine whether *P* and *Q* will collide while moving, but neither of them is allowed to learn any information about the other participant's circle (including its moving path and velocities while moving). In this precondition, Alice and Bob have to check the distance between the two circles endlessly until they detect collision. Our main idea can be expressed as follows:

Protocol 0: Framework of Privacy Preserving Collision Detection of Two Circles

Input: The input of Alice is his private circle *P* and its moving path, and the input of Bob is his private circle *Q* and its moving path.

Output: Both Alice and Bob know whether *P* and *Q* will collide.

Initialization: $t = 0$.

Step 1: Alice and Bob compute the coordinates of the centers of circle *P* and *Q* at time *t* based on their moving paths.

Step 2: Alice and Bob invoke secure protocol <P1> to determine whether *P* and *Q* intersect. If so, output "collides" and quit.

Step 3: Otherwise, Alice and Bob invoke secure protocol <P2> to get the next collision-detection time point t'. Let $t \leftarrow t'$, and go to Step 1.

All we need to do is to implement secure protocols <P1> and <P2> in Protocol 0.

3.2 Secure Intersection Protocol for Two Circles

The relative position of two circles can be determined by comparing the distance between their centers and the sum of their radiuses. We assume neither circle is contained by the other circle, and if the distance between their centers is *less than or equal to* the sum of their radiuses, then the two circles have been collided.

Suppose at some time *t*, the centers of *P* and *Q* are $(x_1, y_1), (x_2, y_2)$ relatively. We can invoke secure scalar product protocol to get the distance between the centers, and the comparison result of the distance and the sum of radiuses can be got by invoking millionaires' protocol.

Protocol 1 Secure Intersection Protocol for Two Circles

Input: The private inputs of Alice are the radius R_p of circle *P* and the current coordinates (x_1, y_1) of its center. Bob's private inputs are R_q and (x_2, y_2) which are the radius of circle *Q* and the current coordinates of its center respectively.

Output: The relative position of *P* and *Q* (intersectant or not).

Step 1: Alice and Bob invoke the second form of secure scalar product protocol, and their private vectors are (x_1, y_1, R_p) and (x_2, y_2, R_q) respectively. After invoking, Alice and Bob respectively get u_1 and u_2 satisfying

$$u_1 = x_1 x_2 + y_1 y_2 + R_p R_q + u_2 \tag{1}$$

Step 2: Alice and Bob invoke millionaires' protocol with input $2u_1 - (x_1^2 + y_1^2 - R_p^2)$ and $2u_2 + x_2^2 + y_2^2 - R_q^2$ respectively. If

$$2u_1 - (x_1^2 + y_1^2 - R_p^2) < 2u_2 + x_2^2 + y_2^2 - R_q^2 \tag{2}$$

then the protocol outputs "not intersectant", otherwise it outputs " intersectant".

Protocol 1 implements the secure protocol <P1> in Protocol 0. In the protocol there are one secure scalar product protocol and one millionaires' protocol performed, therefore the time complexity and communication complexity depend on these two protocols, and can be expressed as $T_{scalar} + T_{comparison}$ and $C_{scalar} + C_{comparison}$ respectively.

Theorem 1: *(Correctness) Protocol 1 can determine the relative position of two input circles, and implements the secure protocol <P1> correctly.*

Proof: We know that the relative position of two circles can be determined by comparing the distance between their centers and the sum of their radiuses. By eq. (1), we can get that ineq. (2) is equivalent to

$$2x_1 x_2 + 2y_1 y_2 + 2R_p R_q - (x_1^2 + y_1^2 - R_p^2) < x_2^2 + y_2^2 - R_q^2 \tag{3}$$

that is,

$$R_p + R_q < \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4}$$

The left of ineq. (4) is sum of two radiuses, and the right is the distance between the centers of two circles. Therefore, if

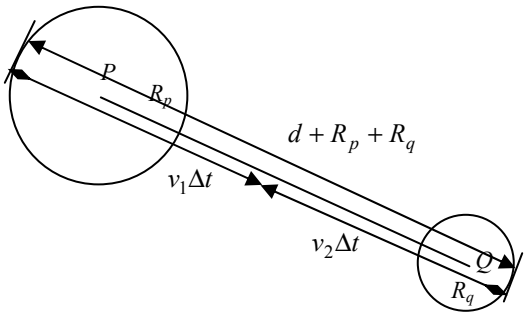

Figure 1 $v_1\Delta t + v_2\Delta t = d + R_p + R_q$

ineq. (4) holds, then the two circles intersect or one circle contains the other. If the latter is the thing, then we can deduce that the two circles intersected some time ago (because in the beginning neither circle is contained by the other circle), and the theorem follows. □

3.3 Secure Protocol for Getting the Next Detection Time Point

There is a naïve scheme to implement secure protocol <P2>: Alice and Bob choose a fixed time interval Δt beforehand, and <P2> returns $t' = t + \Delta t$ when invoked. The advantage of this scheme is its simpleness and low-complexity, and its disadvantage is also obvious: it totally neglects the status information of the circles. In the following a self-adaptive and more elaborate scheme will be presented, which computes a maximal time interval based on the distance and velocities of the two circles. For instance, if the distance between the two circles is long and the speeds are low, the time interval could be longer. Otherwise, the time interval could be shorter.

Suppose in the neighborhood (the definition of the neighborhood will be discussed later) of time point *t*, the maximum velocities of *P* and *Q* are v_1 and v_2 respectively, and suppose the distance of the centers is *d* at time *t*. To detect collision and maximize the time interval between two neighboring time points, we can choose the time interval Δt satisfying that the sum of distances traveled by Alice and Bob is equal to $d + R_p + R_q$ (see *Fig.* 1):

$$v_1\Delta t + v_2\Delta t = d + R_p + R_q \tag{5}$$

That is

$$\Delta t = \frac{d + R_p + R_q}{v_1 + v_2} \tag{6}$$

Because $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, firstly we give the solution for $\sqrt{Z}$ protocol using $(Z + V)$ -disguise scheme.

Protocol 2 $\sqrt{Z}$ -Protocol

Input: Alice knows $Z + V_1$, and Bob knows V_1

Output: Alice gets $\sqrt{Z} + V$, and Bob gets *V*.

Step 1: Bob randomly generates *R* and *V*.

Step 2: Using the $(U \cdot Z)$ -protocol (let the '*V*' in the protocol be zero) on $U = R$, Alice gets $R \cdot Z$.

Step 3: Alice computes $\sqrt{R \cdot Z} = \sqrt{R} \cdot \sqrt{Z}$.

Step 4: Alice and Bob use $(U \cdot Z)$ -protocol on $U = 1/\sqrt{R}$, and this time Alice gets

$$\sqrt{R} \cdot \sqrt{Z} \cdot 1/\sqrt{R} + V = \sqrt{Z} + V$$

and Bob gets *V*.

Now we can present the protocol for securely computing eq. (6).

Protocol 3 Secure Protocol for Getting the Next Detection Time Point

Input: The private inputs of Alice are the radius R_p of circle *P*, the current coordinates (x_1, y_1) of its center at time *t* and the maximum velocity v_1 in the neighborhood of time point *t*. Bob's private inputs are R_q , (x_2, y_2) and v_2 which are the radius of circle *Q* , the current coordinates of its center at time *t* and the maximum velocity in the neighborhood of time point *t* respectively.

Output: The next detection time point t' .

Step 1: Alice and Bob invoke the second form of secure scalar product protocol, and their private vectors are (x_1, y_1) and (x_2, y_2) respectively. Then Alice and Bob respectively get r_1 and s_1 satisfying

$$r_1 = x_1x_2 + y_1y_2 + s_1 \tag{7}$$

Step 2: Alice and Bob invoke $\sqrt{Z}$ protocol, and their private inputs are $x_1^2 + y_1^2 - 2r_1$ and $-(x_2^2 + y_2^2 + 2s_1)$ respectively. Then Alice and Bob respectively get r_2 and s_2 satisfying:

$$r_2 = \sqrt{x_1^2 + y_1^2 - 2r_1 + x_2^2 + y_2^2 + 2s_1} + s_2 \qquad (8)$$

Step 3: Alice computes $r_3 = r_2 + R_p$, and Bob computes $s_3 = s_2 - R_q$.

Step 4: Using Z^{-1}- protocol on v_1 and $-v_2$, Alice and Bob get r_4 and s_4 respectively satisfying:

$$r_4 - s_4 = (v_1 - (-v_2))^{-1} = (v_1 + v_2)^{-1} \qquad (9)$$

Step 5: Using the second form of secure scalar product protocol on (r_3, r_4) and (s_4, s_3), Alice and Bob respectively get r_5 and s_5 satisfying:

$$r_5 = r_3 s_4 + r_4 s_3 + s_5 \qquad (10)$$

Step 6: Alice computes $r_6 = r_3 r_4 - r_5$, and sends it to Bob; Bob computes $s_6 = s_3 s_4 + s_5$, and sends it to Alice.

Step 7: Alice and Bob respectively compute $\Delta t = s_6 + r_6$, and the protocol outputs $t' = t + \Delta t$.

In that way, we implement the secure protocol <P2> in Protocol 0. In Protocol 3 there are two secure scalar product protocols, one Z^{-1} protocol and one $\sqrt{Z}$ protocol performed, therefore the time complexity and communication complexity depend on these three protocols. Z^{-1} protocol and $\sqrt{Z}$ protocol are implemented with two calls of $(U \cdot Z)$ - protocol, so the time complexity and communication complexity of Protocol 3 can be expressed as $2T_{scalar} + 4T_{UZ}$ and $2C_{scalar} + 4C_{UZ}$.

Theorem 2: *(Correctness) Protocol 3 is correct, i.e., the output of Protocol 3* $t' = t + \Delta t$ *satisfies*

$$t' - t = \Delta t = \frac{d + R_p + R_q}{v_1 + v_2} \qquad (11)$$

Proof: By eq. (7) and eq. (8), we can get

$$r_2 - s_2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = d \qquad (12)$$

Because $r_3 = r_2 + R_p$ and $s_3 = s_2 - R_q$ (Step 3), we can get

$$r_3 - s_3 = r_2 + R_p - (s_2 - R_q) = d + R_p + R_q \qquad (13)$$

Combining eq. (10), $r_6 = r_3 r_4 - r_5$ and $s_6 = s_3 s_4 + s_5$ (Step 6), we know

$$r_6 + s_6 = r_3 r_4 - r_5 + s_3 s_4 + s_5$$
$$= (r_3 - s_3)(r_4 - s_4) \qquad (14)$$

Combining eq. (14), eq. (13), eq (9) and the fact that $\Delta t = s_6 + r_6$, we can get that eq. (11) holds. □

Protocol 3 is a general solution that can deal with all kinds of moving paths. On the other hand, it doesn't utilize any information of the moving paths. So it is excellent to handle the situation where the moving paths are undetermined, such as the moving path of an intelligent mobile robot or a spy drone. When the moving path can be determined by a few parameters, such as uniform rectilinear motion, uniform circular motion, free-fall motion, etc., there must be a better solution to do the same job.

The main difficulty for Protocol 3 is the definition of the neighborhood of time point *t*. It can be defined in several ways:

There is no neighborhood if the moving path is undetermined, and the input velocities of Protocol 3 (v_1 and v_2) are estimated maximal velocities based on the past traveling experience. From time *t* to time t', the movement of *P* and *Q* will be controlled not to travel faster than the estimated maximal velocities.

If the moving path is determined beforehand, we can define the neighborhood as a time interval *max_ti*. In this strategy, the input velocities of Protocol 3 (v_1 and v_2) are the maximal velocities between the time *t* and $t + max\_ti$. If the calculated Δt in Protocol 3 is larger than *max_ti*, the output must be $t' = t + max\_ti$. An improved strategy can be implemented by modifying *max_ti* dynamically based on the output of Protocol 3. For example, if the calculated Δt in Protocol 3 is larger than *max_ti*, then we can increase *max_ti*, otherwise, we can decease it. But how to adjust *max_ti* to efficiently decrease the difference between Δt and *max_ti* is not an easy question, and needs a lot of experiments.

3.4 Privacy Preserving Collision Detection of Two Circles

As a template protocol, Protocol 0 can be used to detect collision of two geometric shapes of any types if we can implement the corresponding secure protocol <P1> and <P2>. Protocol 1 and Protocol 3 is designed for the scene where the private geometric shapes are circles. By replacing <P1> and <P2> in Protocol 0 with Protocol 1 and Protocol 3 respectively, we get the protocol that can detect collision of two moving circles.

From the correctness of Protocol 1 and Protocol 3, we can easily deduce the correctness of the protocol:

Theorem 3: *(Correctness) By replacing <P1> and <P2> in Protocol 0 with Protocol 1 and Protocol 3 respectively, we get the protocol that can correctly detect collision of two moving circles.*

Now we give the proof of security.

Theorem 4: *(Security) By replacing <P1> and <P2> in Protocol 0 with Protocol 1 and Protocol 3 respectively, we get the protocol that can securely detect collision of two moving circles.*

Proof: All the data exchanges happen in Protocol 1 and Protocol 3, so we only need to synthesize the exchanged data during the processes of the two protocols to analyze the disclosed information.

We first analyze Protocol 1:

In Step 1, the second form of secure scalar product protocol is invoked, and reveals no information

In Step 2, millionaires' protocol is invoked, and the disclosed information is the predicate:

$$R_p + R_q < \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The disclosed predicate is needed for the problem itself, so Protocol 1 is secure. For Protocol 3, the predicate also do no harm because the precondition of invoking protocol 3 is the predicate itself. So we can ignore the influence of Protocol 1 when analyzing Protocol 3.

In Protocol 3, the results of invoked protocols (including scalar product protocol, Z^{-1} protocol and $\sqrt{Z}$ protocol) are all shared by Alice and Bob. That is, intermediate results don't reveal any information, so the only disclosed information is the protocol's output, i.e. eq. (6).

In summary, the disclosed information for Protocol 0 is eq. (6), which has 4 unknowns (one two-dimensional coordinates, one velocity, one radius). After n iterations of the loop in Protocol 0, each participant can get n equations with $3n+1$ unknowns (n two-dimensional coordinates, n velocities, one radius), and Protocol 0 is also secure. □

It is worth to mention that, the protocol is not secure when the moving path and velocities during traveling can be determined by a few parameters and the type of motion is known by the other participant. In that situation the coordinates and velocities can be represented by these parameters, and after a few iterations of the loop in Protocol 0 the "preserving privacy" promise would be broken. On the other hand, we can find a better solution with much higher efficiency to detect collision in that situation.

4. EXPERIMENTAL RESULTS

We used a small and funny but unreal example. On the equator, a naughty boy throws a stone, and tries to hit a synchronous satellite, see *Fig.* 2. After throwing, the boy wants to know whether the stone and the orbiter collide. There are several strategies to achieve this goal:

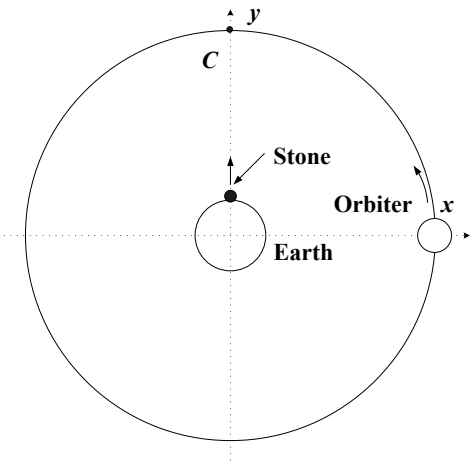

Figure 2 Stone and orbiter

Strategy 1: Every Δt seconds, the distance between the stone and the orbiter are computed in order to check whether they collide (the naïve scheme, see section 3.3).

Strategy 2: The boy uses Protocol 3, and set the neighborhood as a time interval *max_ti* (see section 3.3).

Strategy 3: The boy uses Protocol 3, but this time the time interval *max_ti* is dynamically changed based on the previous output of Protocol 3. There are also several sub-strategies. For example:

Strategy 3.1: If the calculated time interval *ti* is larger than *max_ti*, then set *max_ti* = 1.5×*ti*, otherwise set *max_ti* = *ti*,

Strategy 3.2: After each call of Protocol 3, set *max_ti* = (*ti* + *max_ti*)/2.

The radius of the earth is about 6365km, and the gravity is 9.8m/s^2 (For simplicity, the gravity is unchanged during the rising of the stone). The height of a synchronous satellite is about 35786km, and the period is one day, i.e. 86400s. Suppose the radiuses of the orbiter and stone are 2m and 0.1m respectively. When the stone is thrown, the initial positions are showed in *Fig*. 2, where the center of the orbiter is on the positive x-axis, and the center of the stone is on the positive y-axis. To make them collide on point *C* after 1/4 day, we suppose the initial velocity of the stone is 107497m/s (a very strong boy).

If Strategy 1 is the choice, Δt must be less than the duration of the collision to guarantee that the collision will be detected. Because the duration is about 6.52×10^{-5}s, the count of tests would be 3.31×10^{8}! Comparing to that startling result, the counts of tests for different *max_ti* before the collision is detected are shown in *Table* 1 when other strategies are used. It is obvious that the method of Protocol 3 can greatly reduce the count of tests.

For simplicity, in the experiment the motions of circles are very simple and well-known, which means insecurity (even

Table 1 Counts of tests when different strategies are used

max_ti	Strategy 2	Strategy 3.1	Strategy 3.2
200s	112	17	112
500s	33	16	54
1000s	30	16	40
2000s	22	16	31
5000s	24	16	23
10000s	33	16	19
30000s	77	18	17

when using Strategy 1). But it is inessential, because our aim is to evaluate the effectiveness of our idea for detecting collision of two moving circles.

5. CONCLUSIONS AND FUTURE WORK

This paper addresses the problem of privacy preserving collision detection of two circles, which is one of important computational geometry problems. Four protocols are presented in the paper to solve the problem, and their correctness and security are also analyzed. The experimental results illustrate that our method for detecting collision of two moving circles can greatly reduce the count of tests.

In our future work, we will study the collision detection problems for other types of geometric objects. Other kinds of collision problems, including how to get the collision positions and how to avoid collision while moving, will be studied in the framework of SMC.

6. References

[1] A.C. Yao, Protocols For Secure Computations, In *proceedings of the 23rd Annual IEEE symposium on Foundations of Computer Science*, 1982, pp. 160-164.

[2] Bart Goethals, Sven Laur, Helger Lipmaa and Taneli Mielikäinen. On Private Scalar Product Computation for Privacy-Preserving Data Mining, Available From http://www.adrem.ua.ac.be/~goethals/publications/sspfordm.pdf.

[3] C. Cachin, Efficient Private Bidding and Auctions with an Oblivious Third Party, In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pp. 120-127, 1999.

[4] C. Clifton, M. Kantarcioglu, et al, Tools for Privacy Preserving Distributed Data Mining, In *SIGKDD Explorations*, 4(2): 28-34 December 2002.

[5] I. Ioannidis, A. Grama and M. Atallah, A Secure Protocol for Computing Dot-Products in Clustered and Distributed Environments, In *The 2002 International Conference on Parallel Processing*, Vancouver, British Columbia, Aug. 18-21 2002.

[6] Ioannis Ioannidis and Ananth Grama, An Efficient Protocol for Yao's Millionaires' Problem, In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, 2003.

[7] J. Vaidya and C. Clifton, Privacy Preserving Association Rule Mining in Vertically Partitioned Data, In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 23-26 2002.

[8] Mikhail J. Atallah and Wenliang Du, Secure Multi-Party Computational Geometry, In *Lecture Notes in Computer Science, 2125, Springer Verlag. Proceedings of 7th International Workshop on Algorithms and Data Structures (WADS 2001),* August, 8-10, 2001, Providence, Rhode Island, USA. pp. 165-179.

[9] O. Goldreich, Secure multi-party computation (working draft), Available from http://www.wisdom.weizmann.ac.il/~oded/PS/prot.ps, 1998

[10] S. Goldwasser, Multi-party Computatoin: Past and Present, In *Proceedings of the 16th annual ACM symposium on Principles of distributed computing*, Santa Barbara, CA USA, August 21-24 1997.

[11] Wenliang Du and Zhijun Zhan, A practical Approach to Solve Secure Multi-party Computation Problems, In *Proceedings of New Security Paradigms Workshop*, 2002.