SAFE: A Secure Area of Interests Filter for P2P-based MMOGs

Jianan Hao Suiping Zhou
School of Computer Engineering
Nanyang Technological University, Singapore
haojn@pmail.ntu.edu.sg, ASSPZhou@ntu.edu.sg

***Abstract*—Area of interests (AOI) filter plays a vital role in reducing the network traffic in peer-to-peer (P2P) based massively multiplayer online games (MMOGs). However, not every peer in the system can be completely trusted. Secret information, such as players' positions are vulnerable to be disclosed in traditional AOI filtering and such cheating is hard to detect. We propose a protocol, called SAFE, based on secure computation theory, which is able to keep private data from unauthorized players who may cheat. The proposed SAFE protocol can also be easily implemented based on the existing P2P infrastructure.**

I. Introduction

Over the last decade, Massively Multiplayer Online Game (MMOG) grows rapidly. For illustration, World of Warcraft (WOW) is subscribed by over 11.5 million people on December 2008 [1] compared to 6 million people on January 2006 [2]. The number of players of MMOGs is also expected to increase in the future.

Most MMOGs are based on the Client/Server (C/S) model which facilitates the state control, management and maintenance. However, as the number of players increases, the C/S model cannot perform well due to its poor scalability. To address this concern, several new models are proposed and peer-to-peer (P2P) is considered as a reasonable solution.

Despite using different infrastructure models, most MMOGs employ area of interests (AOI) filters as an efficient technique to eliminate bandwidth consumption on transferring unnecessary information such as invisible updates, the scene behind the mountain, etc. AOI denotes the region that one avatar could directly interact with, for example, it may be defined by the longest distance that one can see or measure. By introducing AOI, one player only receives the events and state updates occurred within the AOI region. Thus, the network traffic will be reduced.

As shown in Figure 1, an AOI filter typically divides the area around an avatar into different regions: an AOI region and an Extension Region (ER). The basic function of AOI filtering is to determine which regions other avatars are located with respect to a local avatar. ER is used to adjust AOI filtering frequency. For instance in Figure 1, let us consider Alice (A) who is performing AOI filtering. The dark grey region represents her AOI, and the light grey region represents her ER. In this case, Bob (B) is inside Alice's AOI, thus every state updates of Bob will be sent to Alice. Carol (C) is outside Alice's AOI but inside her ER, thus Alice needs to perform AOI filtering with Carol frequently. Dave (D) is outside Alice's ER, thus Alice needs to invoke AOI filtering with Dave less frequently.

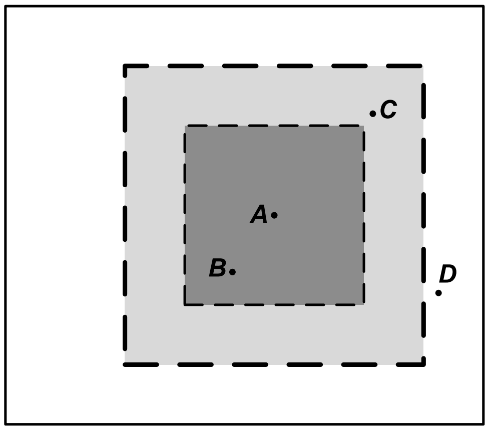

Figure 1. AOI and Extension Region

As position information of an avatar is important for many MMOGs, in general, it should only be sent to other avatars who are within the AOI of this avatar. However, to perform AOI filtering, a local avatar needs to know the positions of other avatars regardless of their current positions. Therefore, we face a dilemma in AOI filtering between "one should know the positions of others only when they are within his/her AOI" and "AOI filtering requires the positions of others whether or not they are within the avatar's AOI".

In C/S model, this difficulty in AOI filtering is addressed by relying on trusted centralized server. However, in P2P model, no peer can be completely trusted. Disclosing the current position of an avatar before it enters the AOI of another avatar may give the latter unauthorized advantages in the game. For example, the latter avatar may prepare an ambush of the former avatar based on the knowledge of the current position of the former.

Unlike some widely studied types of cheating such as suppressed update, fixed delay, inconsistency and undo [3] which are all active attacks, information exposure caused by AOI filtering is a kind of passive attacks that can not be easily detected. For example, to defeat an inconsistency attack, we can have a voting protocol to recover incorrect states as well as to detect the malicious peer. However, for AOI filter, we cannot even know whether information is disclosed because an attacker can behave exactly the same as normal peers but peek others' positions in local memory. More importantly, once the information is revealed, additional effort on recovery is useless because disclosed secret may have already been applied to gain unfair benefits in the game.

The RACS system [4] employs referee from trusted nodes to perform AOI filtering. However, its hybrid architecture alters the original P2P infrastructure. It also creates new problems such as referee election and extra latency which may diminish the benefit of P2P overlay. Other anti-cheat systems [5]–[7] mainly rely on majority voting. Unfortunately, although it is efficient to solve consistency problem, majority voting is helpless to information exposure attacks.

In this paper, we ask a fundamental question: is it possible to perform AOI filtering in P2P model without disclosing the current position of an avatar to other peers who are not supposed to know this information? We propose a new protocol—Secure Area of interests FiltEr (SAFE) to avoid information exposure during AOI filtering. It can be shown that for cases where cheaters are in the majority, SAFE can protect peers' private data from being revealed to authorized peers. Attacks that can be defeated by SAFE include but not limited to eavesdropping and modifying network packets, peeking in and tempering local memory in addition to deep reverse engineering [8]. Besides, as SAFE is not based on majority voting but on the basis of secure computation in cryptography theory, it guarantees the computational infeasibility to recover secret information, i.e., player's position. It also avoids inefficient synchronization on voting results. Moreover, SAFE only overrides traditional AOI filtering algorithms, thus transparent to higher level hierarchy.

The rest of this paper is organized as follows. In section II, we first analyze how information revelation may occur in AOI filtering. In section III, we introduce SAFE and describe its assumptions. Section IV briefly reviews secure computation as a basis of SAFE. Section V presents the details of SAFE as well as analysis. We also evaluate related work in Section VI. Finally, section VII concludes the paper.

II. Security Issues in AOI Filtering

AOI filtering is firstly invented as a technique to reduce processor and bandwidth consumption in MMOG. At the time when P2P model is not prevalent, C/S model dominates MMOG design. Server has global view of peers and can perform AOI filtering safely. However, when MMOG comes to P2P model, storage in P2P-based MMOG is distributed in untrusted nodes. To perform AOI filtering, two avatars' positions must be provided as input parameters prior to computation. In other words, in order to employ AOI filtering, secrets such as avatars' positions must be revealed first. Although such untitled information will not be shown to players as designed, it is very likely to peek that data in local memory. In many genres of MMOG, position is treated as a very critical information of tactics. By knowing others' positions, unjust benefits will be obtained and it will further reduce the subscriber's interests.

In this section, we will first show the general algorithm used in P2P-based AOI filtering and examine its weakness. Later, we will analyze in detail how the algorithm could result in information exposure attack and how this could affect the fairness of games.

Figure 2 illustrates the general AOI filtering algorithm. Hereinafter, we assume that Alice (one peer) is the peer who performs the AOI filtering to determine whether Bob (another peer) is in her AOI region, ER region or outside ER region. The algorithm also adjusts the interval of how frequent the AOI filtering is performed.

A core function of the algorithm is **GetRegion(*Alice*, *Bob*)**, which is shown in Figure 3. After obtaining the value of *Region*, Alice will establish direct connection with Bob only if he is inside her AOI; otherwise she will disconnect existing connection with him. The interval for next enquiry is also determined by the value of *Region*. It will be set to *Low* if Bob is far from her indicated by *Region* equal to *OutsideER*. In other cases, Alice has to repeat enquiry frequently by setting *Interval* to *High*.

A. P2P-based AOI Filter

```
/*Alice's AOI filter on Bob*/
loop
    $Region$=GetRegion($Alice$,$Bob$)
    if $Region = InsideAOI$ then
        Connect($Bob$)
        $Interval = High$
    else if $Region = InsideER$ then
        Disconnect($Bob$)
        $Interval = High$
    else if $Region = OutsideER$ then
        Disconnect($Bob$)
        $Interval = Low$
    end if
    Wait($Interval$)
end loop
```

Figure 2. General AOI filtering

```
/*GetRegion($Alice$,$Bob$)*/
Update($Bob.Pos$)
$Distance$=GetDistance($Alice.Pos$,$Bob.Pos$)
if $Distance <= AOIRange$ then
    return $InsideAOI$
else if $Distance <= ERRange$ then
    return $InsideER$
else
    return $OutsideER$
end if
```

Figure 3. GetRegion algorithm

$AOIRange$ and $ERRange$ in Figure 3 generally represent the radius of AOI and ER respectively. The **GetRegion** algorithm is vulnerable to information exposure attack since it incorrectly assumes the peer is trustworthy. In line 2, it first communicates with Bob to retrieve his position. At this point, Bob's position is stored in Alice's local memory which may be accessible by malicious codes on her machine.

B. Information Exposure

In general, information exposure attack results in biased information for different players who are supposed to response just according to visual and auditory information designed by game developers. For most games, screen is the main source that they receive information from the game. One may examine his health, mana and items from status label, or instead try to aim an enemy to prepare an attack. All these information are designed to be fair among players by well-known rules of the game. For example, as a limitation to sight, avatars may not see objects including opponents beyond a certain distance. Or he cannot know whether there is a threat behind the stone. These rules will finally evolve to tactics such as lurking to assault as well as corresponding defence strategy.

However, if someone gains untitled information beyond these rules, the balance would be broken. For instance, hiding behind a wall should be considered as a good strategy for ambushing. But if a cheater can see through the wall by "wallhack" [9], he is possible to discover the trap and avoid it. Even worse, he may perform tit-for-tat operation, e.g., throwing grenade to explode the ambush. Once a player suspects someone is cheating, he may request help from game publisher who acts as an arbitrator to determine whether cheating occurred. Unfortunately, information exposure leaves less or even no evidence since these cheats are done silently. For example, "wallhack" attack is rather easy to achieve by simply disabling depth test to render hiding objects. The malicious code can be inserted into game program or more clever, as a part of graphics card driver. As a result, most judgements on cheats are based on analyzing the behaviors of player who is over-lucky. But this method suffers in distinguishing a skilled gamer from cheaters. In addition, the analysis is tedious and costly.

It is worthy to note that information exposure attack belongs to passive attack which can not be efficiently detected. For example, one can just extract others' positions from memory without interfering the game. From the view of other peers, the malicious peer behaves just as normal. Moreover, even assuming the attack can be detected by advanced identification technique to distinguish an expert player and a cheating player, the damaging effect is unrecoverable once the secret is disclosed. Unlike active attacks, e.g., inconsistency attack, which can be fixed by amending the states later after cheating detection, the only way to defeat information exposure attack is to avoid it at the beginning.

III. Problem Definition, Assumptions & Attack Model

Before we present our solution to information exposure attack, we will first give the basic requirements of SAFE and assumptions that have been made in this paper. We will also discuss various attacks related to AOI filtering.

A. Basic Requirements on SAFE

To perform AOI filtering and also protect position information from being disclosed to unauthorized peers, SAFE must provide the following three properties:

- **P1:** During AOI filtering, positions of Alice and Bob are never revealed to each other, nor to a third party.
- **P2:** When Alice and Bob both faithfully carry out the protocol, they can get correct results.
- **P3:** Alice and Bob can verify the results to defeat connection cheating.

Property **P1** guarantees information exposure attack is prevented. In addition, any third party is not involved in SAFE to avoid dependency on trusted nodes. **P2** ensures SAFE maintains correct functionality to perform AOI filtering. Alice and Bob should get correct results if they fulfill the protocol.

It is worthy to note that when malicious code is installed on platform, SAFE may not output correct result. Although this will not reveal the position information, it leads to wrong judgement on whether to set up a direct connection. Since more secrets will be exchanged after direct connection is established, an attacker may cheat another peer by simply announcing "I am inside your AOI". In this paper, we use *connection cheating* to denote this attack. To prevent it, additional mechanisms are provided to check the authenticity of the answer received. That is also the reason why **P3** is necessary.

B. Assumptions

To implement SAFE, we firstly assume every peer holds a pair of identification keys and their public parts are well distributed such as in public key infrastructure (PKI). Note that creation and distribution of keys are fundamental processes on session initialization for major games.

Without loss of generality and for simplicity, we assume the AOI regions and ER regions have square shapes, although other shapes such as circles are also possible to represent these regions. We also assume the position of an avatar is represented in a 2D space.

It should be pointed out that SAFE neither assumes a trusted third-party nor assumes that honest peers are in the majority. Lastly, a reputation system is not required which means that SAFE does not depend on the analysis of the historical behavior of players.

C. Attack Modeling

In this section, we will define the behavior of attackers, including their capabilities and attacking methods.

An *attacker* refers to a peer who wants to obtain untitled information—the positions of other *honest players* who will faithfully play the game. We do not set a limit to the number of attackers. However, for a meaningful discussion, we assume that at least one player in the game is honest.

For the ability of an attacker, it generally includes, but not limited to: 1) peeking and modifying data stored in local memory; 2) tracking and altering code in local memory; 3) eavesdropping packets on network; 4) collusion with other attackers by communication via second channel. In other words, the attacker can do everything locally and is free to communicate with other attackers.

Nevertheless, we assume that an attacker is computationally constrained. Moreover, an attacker cannot affect other peers via channels outside the game itself. It means that an attacker cannot physically or logically control other peers' computers. For instance, the attacker has no ability to inject malicious programs to remote computer nor peek data stored remotely. We define the term *cost of attack* as what attacker needs to pay to achieve his attack, e.g., the time needed for sniffing packets on network or decompiling the source code. On the other hand, *benefit of attack* refers to what profit an attacker can obtain from the attack. An effective attack can be achieved by raising the benefit of attack while reducing the cost of attack.

The potential attacks involve 1) establishing query table by pre-computation attack; 2) mining information to speculate private data; 3) bypassing the system. We will give the corresponding solutions in section V.

IV. Secure Computation

To our knowledge, existing solutions to enhancing security of MMOG do not employ secure computation. Secure computation is a classical concept which was introduced by A. Yao in [10] and related to oblivious transfer in [11]. Essentially, "secure computation" refers to situation that multiple parties plan to compute a value determined by inputting the secret numbers held by individuals but do not wish to disclose their secrets to one another.

One special case is that 1) only two parties participate in the computation; 2) the result is a boolean value, *TRUE* or *FALSE* ; 3) the input values are integer variables of bounded range. This simple case can be illustrated by Yao's millionaires's problem [10].

The problem is that two millionaires want to know who is richer but do not want to reveal the exact wealth information to each other. Let us suppose Alice has a millions and Bob has b millions, where $1 \leq a, b \leq M$. Then we will need a protocol to decide whether $a \leq b$ which is also the only thing they will know finally.

Secure computation can also be applied to other scenarios, e.g., secret voting [10], oblivious negotiation [10], etc. For SAFE, we choose secure computation as a key to solve the dilemma in AOI filtering that we have pointed out in the Introduction.

V. Methodology and Algorithms

In this section, we present the algorithms of SAFE in detail. First, we describe how to transform the AOI filtering problem into a secure computation problem. Next, we modify

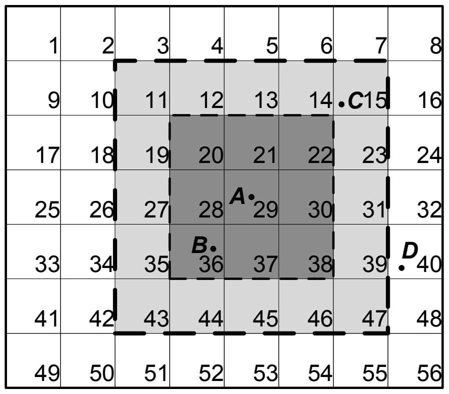

Figure 4. Virtual world represented by primitives

```
1:  /*GetRegion($Alice$,$Bob$)*/
2:  $AliceAOI$ = AOI($Alice$)
3:  $IsInsideAOI$ = INSIDE($AliceAOI$, Pos($Bob$))
4:  **if** $IsInsideAOI$ **then**
5:      **return** $InsideAOI$
6:  **else**
7:      $AliceER$ = ER($Alice$)
8:      $IsInsideER$ = INSIDE($AliceER$, Pos($Bob$))
9:      **if** $IsInsideER$ **then**
10:         **return** $InsideER$
11:     **else**
12:         **return** $OutsideER$
13:     **end if**
14: **end if**
```

Figure 5. Modified GetRegion algorithm

the original **GetRegion** algorithm into a secure-computation compatible version. A key function—INSIDE in this revised **GetRegion** algorithm will be described. We also discuss why SAFE can defeat attacks mentioned previously. Finally, we discuss how to defeat connection cheating.

A. Dividing the Virtual World into Primitives

To convert the AOI filtering problem into a secure computation problem, we need to discretize the virtual world into a set of primitives. As shown in Figure 4, the virtual world is divided into 56 primitives and each of them is assigned by a distinct number, namely *cell number*. Again, the dark grey region illustrates AOI and the light grey region represents ER. Let $\mathrm{Pos}(x)$ denote the cell number where player x resides and I refer to the universal set of cell numbers in virtual world. In this case, $I = \{1, 2, 3, ..., 56\}$, $\mathrm{Pos}(A) = 29$, $\mathrm{Pos}(B) = 36$, $\mathrm{Pos}(C) = 15$ and $\mathrm{Pos}(D) = 40$. Symbol $\mathrm{AOI}(x)$ and $\mathrm{ER}(x)$ are introduced to express the set of cell numbers for AOI and ER representation. For instance, $\mathrm{AOI}(A) = \{20, 21, 22, 28, 29, 30, 36, 37, 38\}$ and $\mathrm{ER}(A) = \{11, 12, 13, 14, 15, 19, 23, 27, 31, 35, 39, 43, 44, 45, 46, 47\}$.

B. Modified **GetRegion** algorithm for AOI filtering

Figure 5 shows the modified **GetRegion** algorithm for AOI filtering. *AliceAOI* and *AliceER* refer to Alice's AOI/ER region represented by sets of cells. Boolean variables *IsInsideAOI* and *IsInsideER* indicate whether Bob is inside Alice's AOI/ER returned by a function $\text{INSIDE}(S, b)$ defined in Equation 1. Note that S is known to Alice but b is a secret held by Bob remotely. When invoke the function, b is input by a reference name, e.g., $\text{Pos}(Bob)$.

$$\text{INSIDE}(S, b) = \begin{cases} \textit{TRUE} & \exists i \in S \Rightarrow i = b \\ \textit{FALSE} & \text{otherwise} \end{cases} \tag{1}$$

C. Algorithm for $\text{INSIDE}(S, b)$

Let $|S|$ be the cardinality of set S. Figure 6 shows the algorithm for $\text{INSIDE}(S, b)$.

```
Require: Alice holds $S = \{a_1, a_2, a_3, ...\}$ and Bob holds $b$.
    A pair of asymmetric keys is finely distributed and only
    Bob knows the private part.
Ensure: Alice and Bob both know $\text{INSIDE}(S, b)$.
1: Alice chooses large and random integers $x_i$, and encrypts
   it by Bob's public key. $c_i = \text{Encode}(x_i)$, $1 \le i \le |S|$.
2: Alice computes $y_i = c_i - a_i$, $1 \le i \le |S|$ and sends the
   result to Bob.
3: Bob calculates $z_i = \text{Decode}(y_i + b)$, $1 \le i \le |S|$.
4: Bob chooses a large prime $p$, preferably smaller than $x$.
   Then Bob computes $w_i = (z_i \bmod p)$, $1 \le i \le |S|$.
5: Bob scrambles $w_i$ and then gets $u_i$. Bob sends $u_i$ and $p$
   to Alice.
6: Alice checks whether $\exists 1 \le i, j \le |S| \Rightarrow x_i \equiv u_j$
   $(\bmod\ p)$. If so, she concludes $b$ belongs to $S$; If not, $b$
   does not belong to $S$.
7: Alice tells Bob her conclusion.
```

Figure 6. INSIDE algorithm

During the execution, Alice and Bob interact by carrying out the protocol. In general, the lengths of x_i and p are predefined though their values are generated secretly on each side. Moreover, the domain of x_i is supposed to be very large, e.g., 2^{64} to defeat pre-computation attack. Although any existing asymmetric encryption algorithm can be used, we prefer the algorithm, e.g., RSA [12] that provides good diffusion among x_i, $\text{Encode}(x_i)$ and $\text{Decode}(x_i)$ for better security strength. In addition, when Bob scrambles w_i at step 5, he should ensure that the permutation is unbiased. We will now discuss how this protocol satisfy the properties defined in section III-A.

Remark 1. *If Alice and Bob faithfully execute the protocol,* $\text{INSIDE}(S, b)$ *can always output* TRUE, *if* $b \in S$.

Proof: First, w_i can be expanded according to the protocol.

$$\begin{aligned} w_i &= z_i \bmod p \\ &= \text{Decode}(y_i + b) \bmod p \\ &= \text{Decode}(c_i - a_i + b) \bmod p \\ &= \text{Decode}(\text{Encode}(x_i) - a_i + b) \bmod p \end{aligned}$$

Since b belongs to S, without loss of generality, let $a_k = b$ where $1 \le k \le |S|$ and $u_{k'} = w_k$ after Bob scrambled w at step 5.

$$\begin{aligned} u_{k'} &= w_k \\ &= z_k \bmod p \\ &= \text{Decode}(y_k + b) \bmod p \\ &= \text{Decode}(c_k - a_k + b) \bmod p \\ &= \text{Decode}(\text{Encode}(x_k) - a_k + b) \bmod p \\ &= \text{Decode}(\text{Encode}(x_k)) \bmod p \\ &= x_k \bmod p \end{aligned}$$

Therefore, the answer will be *TRUE* according to step 6 where $i = k$ and $j = k'$. ■

Remark 2. *If Alice and Bob faithfully execute the protocol,* $\text{INSIDE}(S, b)$ *can be expected to output* FALSE *if* $b \notin S$.

Proof: Assuming $\forall a \in S \Rightarrow a \ne b$, but $\text{INSIDE}(S, b)$ outputs *TRUE* because $x_i \equiv u_j \pmod p$. Without loss of generality, let $u_j = w_{j'}$ after Bob scrambled w at step 5.

$$x_i \equiv u_j \equiv w_{j'} \pmod p$$

$$\begin{aligned} x_i &= w_{j'} + tp \text{ , where } t \in \mathbb{Z} \\ tp &= x_i - w_{j'} \\ t &= \frac{x_i - w_{j'}}{p} \\ &= \frac{x_i - \text{Decode}(\text{Encode}(x_{j'} - a_{j'} + b))}{p} \end{aligned}$$

It implies that $x_i - \text{Decode}(\text{Encode}(x_{j'} - a_{j'} + b))$ is dividable by p. If given the fixed p, noted that $a_{j'}$ and b is already determined and not identical, finding x_i and $x_{j'}$, if any, which satisfies the formula could be computationally infeasible only if p is large enough. Actually, as Alice and Bob randomize x_i and p separately in the protocol, the possibility of satisfying the formula is negligible. Therefore, the assumption is hard to reach and original proposition can be proved. ■

According to Remark 1 and Remark 2, **P2** is satisfied.

For **P1**, first it is easy to see SAFE is executed between two parties and secrets are never disclosed directly. Even if the result is *TRUE*, Alice cannot know which primitive Bob locates at step 6. Since Bob hashes z_i into w_i at step 4 and further permutate w_i into u_i at step 5, link between x_i and u_i is cut due to compression and permutation. As $\forall 1 \le i, j \le |S| \Rightarrow Pr(a_i = b) = Pr(a_j = b)$, SAFE never tells Alice where Bob is. For an attacker, one possible method to recover b is to re-establish link between x and u.

Noted that the protocol does not depend on a secure communication channel and no messages are encrypted to prevent from eavesdropping, it is unnecessary to study another third-party. In fact, Alice and Bob are two peers who have the most knowledge to reveal the secrets.

Remark 3. *If Alice cheats in* $\mathrm{INSIDE}(S, b)$ *but Bob faithfully execute the protocol, Alice still cannot know* b.

Proof: If assuming Alice cheats, she may deliberately select y_i in step 2. To reverse the compression effect of mod function at step 4 where original z_i lost, Alice will expect $w_i = z_i$ at step 3. To achieve it, she will assign $x_i = e_i$ where $e_i < p$ and set $a_i = b'_i$ where b'_i are estimates of b. If she guesses right, $u_{i'} = w_i = e_i$ will be received at step 4 and she knows $b'_i = b$; otherwise, she knows $\forall k \in |S| \Rightarrow a_k = b'_k \neq b$. Without further information on estimation, for each time SAFE is executed,

$$Pr(hit) = Pr(\exists k \in |S| \Rightarrow a_k = b'_k = b) = \frac{|S|}{|I|}$$

Since SAFE only gives binary answer to Alice, if she got *FALSE*, there is no further hints on the deviation between b and b'_i. Alice can only gamble on Bob's position. $Pr(hit)$ is a relatively constant value determined by the property of the game. When game world goes large or AOI/ER goes small, the possibility decreases. We also suggest Bob should check the z_i at step 4 and deny Alice's enquiry if $z_i < p$.

In some cases, Alice can also employ additional information to estimate b. For example, she may know Bob located on her left side before a short while. Then she may skip testing primitives on her right and thereby increase $Pr(hit)$. However, it also can be defeated by verifying $z_i < p$ at step 4.

Another kind of attacks is to crack the encryption architecture, e.g. recovering Bob's private key. If Alice successfully obtains Bob's private key, pre-computation attack is possible. She may send Bob selected y_i at step 2 and Bob will compute $z_i = \mathrm{Decode}(y_i + b)$. Because domain of b is limited, it is feasible to illustrate all pairs of $(b, y_i, \mathrm{Decode}(y_i + b))$ as a constant table with space complexity of $O(|I||S|)$. It is quite feasible for modern computer to store it. Although z_i is not directly disclosed to Alice, given query table, it is possible to recover it from u_i and p. In this way, corresponding b can also be found. However, this kind of attack can be easily prevented by choosing a reliable encryption function.

To sum up, Alice cannot recover Bob's b if 1) game world is large enough compared to AOI/ER region, or Bob can deny Alice's query with unsafe z_i; 2) encryption function used in protocol is secure. ■

Remark 4. *If Bob cheats in* $\mathrm{INSIDE}(S, b)$ *but Alice faithfully execute the protocol, Bob still cannot know* S.

Proof: If assuming Bob cheats, he will expect to recover at least one element of S—a_i from y_i sent by Alice at step 2. Given $y_i = \mathrm{Encode}(x_i) - a_i$, Bob has no direct information to a_i.

However, since Bob knows the public part of the key pair for $\mathrm{Encode}()$, pre-computation attack can be done by treating y_i as a one-way function with x_i and a_i for inputs. To achieve it, he has to construct a table storing all y_i for every possible combination of x_i and a_i. A possible table heading would be y_i, x_i and a_i where x_i can be dropped after construction. After that, once he received a y_i, he will search the table to get the row with the same value and find out corresponding a_i. By doing this for each y_i, it is possible to recover each a_i and construct the whole S.

Nevertheless, this attack can be defeated if Alice chooses x with large domain. Let M refer to domain of x and therefore the space complexity of table is $O(|M||S|)$. If $|M|$ is large enough, e.g., 2^{64}, it is computationally infeasible for Bob to fully store this table. In this case, pre-computation attack cannot be achieved effectively. ■

Consequently, **P1** is also satisfied according to proofs above.

D. Mutual Inspection

We have proved SAFE itself will not disclose secret. However, since the protocol is carried out between two parties, we cannot guarantee Alice and Bob both fulfill the contract. For example, Alice firstly knows the answer at step 6 but she may not tell the truth to Bob at the next step. As an initiator, Alice can randomly generate y_i at step 2 and send it to Bob. Instead of doing computation on u_i and p at step 6, she can directly tell Bob that he is within her AOI. The wrong answer will lead to connection cheating. Although this issue is out of the boundary that AOI filtering covered, we will discuss its possible solutions to mitigate it.

We propose to use mutual inspection to solve this problem. Mutual inspection is a notation for bidirectional verification. The motivation is that answer Alice announced is not fully trusted by Bob. In this case, Bob will initiate the protocol again which looks like he exchanges his character with Alice. Because this time Bob will first know the answer, thus protecting him from connection cheating.

For instance, in phase 1 Alice tells Bob he is within her AOI but actually not. Later, Bob initiates the protocol and Alice acts as a responser in phase 2. If Alice wants Bob to get *TRUE* at step 6, she has to guess a cell number which belongs to Bob's AOI. As discussed above, it is difficult to achieve. Bob therefore can detect Alice was lying.

By employing mutual inspection, **P3** is satisfied. Initiator can hardly sabotage the protocol and thereby cannot complete connection cheating attack. However, mutual inspection introduces additional iterations of interaction that may increase latency. A tradeoff between security and performance should be evaluated for a certain case.

VI. Related Work

The Referee Anti-Cheat Scheme (RACS) is proposed by Webb *et al.* [4] as a solution to enhance the scalability of C/S model without decreasing security. We studied RACS mainly because it provides a mechanism to defeat information exposure attack.

RACS is a hybrid architecture of C/S and P2P. In P2P mode (it is called PP mode in RACS), it allows peers to communicate directly and thus improves its scalability. To prevent cheats, the referee whose role is similar to server in C/S model acts as an arbitrator to validate peers' behaviors. If referee suspects a peer is cheating, it will perform reversion on that peer from PP mode to PRP (Peer-Referee-Peer) mode which only allows the peer exchange updates with the referee. To eliminate information exposure, RACS confines secret information to the transmissions between peers and referee. In other words, the referee has to receive and simulate every peers' updates vulnerable to disclose. The main difference between PP and PRP is that the former offloads simulation of unsensitive updates onto peers while the latter requires that referee handles every update like server in C/S model.

The experimental data shows that RACS reduces delay and referee's outgoing bandwidth compared to existing solutions. But the author also points out RACS is powerless to address scalability issues of incoming bandwidth and the referee processing limitation. In addition, confining the referee to one trusted node may not stand on the point with optimal latency where is an important benefit of P2P-based MMOG. If distributing referee to normal peers is allowed, it will create new issues like referee selection, synchronization, etc. which are not discussed in [4]. Lastly, as secret information can be exchanged only by referee, the referee also becomes a single point of failure.

VII. Conclusion and Future Work

We have analyzed how AOI filtering is vulnerable to information exposure attack and proposed SAFE as a solution. SAFE has the following major benefits: 1) it does not require honest peers are in the majority; 2) no trusted node or server is involved to fulfill the protocol; 3) it is compatible to most existing AOI filter designs; 4) it is resistant to connection cheating.

Admittedly, SAFE also has its limitations. For example, the secure computation involved requires encryption operations that are heavy computation tasks. We are investigating how these overheads affect the effectiveness of the system and their possible solutions. We also plan to extend the mechanisms used in SAFE to other components of MMOGs besides AOI filtering.

References

[1] B. Entertainment., "World of warcraft subscriber base reaches 11.5 million worldwide," http://us.blizzard.com/en-us/company/press/pressreleases.html?081121, 2008.

[2] MMOGCHART.com, "MMOGCHART.com," http://www.mmogchart.com/, 2008.

[3] S. D. Webb and S. Soh, "A survey on network game cheats and p2p solutions," *Australian Journal of Intelligent Information Processing Systems*, vol. 9, no. 4, pp. 34–43, 2007 (Published 2008).

[4] S. D. Webb, S. Soh, and W. Lau, "RACS: a referee anti-cheat scheme for P2P gaming," in *NOSSDAV'07: 17th International workshop on Network and Operating Systems Support for Digital Audio & Video*, 2007, pp. 37–42.

[5] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low latency and cheat-proof event ordering for peer-to-peer games," in *NOSSDAV '04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA: ACM, 2004, pp. 134–139.

[6] M. Gorawski and K. Stachurski, "A Secure Event Agreement (SEA) protocol for peer-to-peer games," in *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 34–41.

[7] A. B. Corman, P. Schachte, and V. Teague, "A Secure Group Agreement (SGA) Protocol for Peer-to-Peer Applications," *Advanced Information Networking and Applications Workshops, International Conference on*, vol. 1, pp. 24–29, 2007.

[8] I. V. McLoughlin, "Secure Embedded Systems: The Threat of Reverse Engineering," in *ICPADS*, 2008, pp. 729–736.

[9] J. Kücklich, "Wallhacks and aimbots," *Space Time Play: Computer Games, Architecture and Urbanism: the Next Level*, pp. 118–121, 2007.

[10] A. C. Yao, "Protocols for secure computations," in *SFCS '82: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1982, pp. 160–164.

[11] M. O. Rabin, "How to exchange secrets with oblivious transfer," Cryptology ePrint Archive, Report 2005/187, 1981, http://eprint.iacr.org/.

[12] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.