TRUST MANAGEMENT WITHOUT REPUTATION IN P2P GAMES

Adam Wierzbicki
Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, Warsaw, Poland

Keywords: Peer-to-peer computing, trust management, massive multiplayer online games, secret sharing.

Abstract: The article considers trust management in Peer-to-Peer (P2P) systems without using reputation. The aim is to construct mechanisms that allow to enforce trust in P2P applications, where individual peers have a high possibility of unfair behaviour that is strongly adverse to the utility of other users. An example of such an application of P2P computing is P2P Massive Multi-user Online Games, where cheating by players is simple without centralized control or specialized trust management mechanisms. The article presents new techniques for trust enforcement that use cryptographic methods and are adapted to the dynamic membership and resources of P2P systems.

1 INTRODUCTION

The Peer-to-peer (P2P) computing model has been widely adopted for file-sharing applications. Other examples of practical use of the P2P model include distributed directories for applications such as Skype, content distribution or P2P backup. Clearly, the P2P model is attractive for applications that have to scale to very large numbers of users, due to improved performance and availability. However, using the P2P model for complex applications still faces several obstacles. Since the P2P model requires avoiding the use of centralized control, it becomes very difficult to solve coordination, reliability, security and trust problems. A large body of ongoing research aims to overcome these problems and has succeeded in some respects. It remains to be shown whether the results of this research can be applied to build complex applications using the P2P model.

It is the aim of this paper to consider how the P2P model could be applied to build a Massive Multiplayer Online (MMO) game. At present, scalability issues in MMO games are usually addressed with large dedicated servers or even clusters. According to white papers of a popular multi-player online game – TeraZona (Zona, 2002) – a single server may support 2000 to 6000 simultaneous players, while cluster solutions used in TeraZona support up to 32 000 concurrent players. The client-server approach has a severe weakness, which is the high cost of maintaining the central processing point. Such an architecture is too expensive to support a set of concurrent players that is by an order or two orders of magnitude larger than the current amounts. To give the impression of what scalability is needed – games like Lineage report up to 180 000 concurrent players in one night.

MMO games are therefore an attractive application of the P2P model. On the other hand, MMO games are very complex applications that can be used to test the maturity of the P2P model. In a P2P MMO game, issues related to trust become of central importance, as shall be shown further in this paper. How can a player be trusted not to modify his own private state to his advantage? How can a player be trusted not to look at the state of hidden objects? How can a player be trusted not to lie, when he is accessing an object that cannot be used unless a condition that depends on the player's private state is satisfied? In this paper, we show how all of these questions can be answered. We also address performance and scalability issues that are a prime motivation for using the P2P model. For the first time, an integrated architecture for security and trust in P2P MMO games has been developed.

Our trust management architecture does not use reputation, but relies on cryptographic mechanisms that allow players to enforce trust by verifying fairness of moves. Therefore, we call our approach to trust management "trust enforcement". The trust

Wierzbicki A. (2006).
TRUST MANAGEMENT WITHOUT REPUTATION IN P2P GAMES.
In *Proceedings of the International Conference on Security and Cryptography*, pages 126-134
DOI: 10.5220/0002104901260134

management architecture proposed for P2P MMO games makes used of trusted central components. It is the result of a compromise between the P2P and client-server models. A full distribution of the trust management control would be too difficult and too expensive. On the other hand, a return to the trusted, centralized server would obliterate the scalability and performance gains achieved in the P2P MMO game. Therefore, the proposed compromise tries to preserve performance gains while guaranteeing fairness of the game. To this end, our trust management architecture does not require the use of expensive encryption, which could introduce a performance penalty.

In the next section, security and trust issues in P2P MMO games are reported and illustrated by possible attack scenarios. In section 3, some methods of trust management for P2P MMO games will be proposed. Section 4 presents a security analysis that demonstrates how the reported security and trust management weaknesses can be overcome using our approach. Section 5 discusses the performance of the presented protocols. Section 6 describes related work, and section 7 concludes the paper.

2 SECURITY ISSUES IN P2P MMO GAMES

The attacks described in this section illustrate some of the security and trust management weaknesses of P2P game implementations so far. We shall use a working assumption that the P2P MMO game uses some form of Dynamic Hash Table (DHT) routing in the overlay network, without assuming a specific protocol. In the following section, we describe a trust management architecture that can be used to prevent the attacks described in this section.

Private state: Self-modification

P2P game implementations that allow player to manage their own private state (Knutsson, B., 2004) do not exclude the possibility that a game player can deliberately modify his own private state (e.g. experience, possessed objects, location, etc.) to gain advantage over other game players. A player may also alter decisions already made in the past during player-player interaction that may affect the outcome of such an interaction.

Public state: Malicious / illegal modifications

In a P2P MMO game, updates of public state may be handled by a peer who is responsible for a public object. The decision to update public state depends then solely on this peer – the coordinator. Furthermore, the coordinator may perform malicious modifications and multicast illegal updates to the group. The falsified update operation may be directly issued by the coordinator and returned back to the group as a legal update of the state. Such an illegal update may also be issued by another player that is in a coalition with the coordinator, and accepted as a legal operation.

Attack on the replication mechanism

When state is replicated in a P2P game, replication players are often selected randomly (using the properties of the overlay to localize replicated data in the virtual network). This can be exploited when the replication player can directly benefit from the replica of the knowledge he/she is storing (i.e. the replication player is in the region of interest and has not yet discovered the knowledge by himself).

Attack on P2P overlays

In a P2P overlay (such as Pastry), a message is routed to the destination node through other intermediary nodes. The messages travel in open text and can be easily eavesdropped by competing players on the route. The eavesdropped information can be especially valueable if a player is revealing his own private state to some other player (player–player interaction). In such case, the eavesdropping player will find out whether the interacting players should be avoided or attacked.

The malicious player may also deliberately drop messages that he is supposed to forward. Such an activity will obstruct the game to some extent, if the whole game group is relatively small.

Conclusion from described attacks

Considering all of the attacks described in this chapter, a game developer may be tempted to return to the safe model of a trusted, central server. The purpose of this article is to show that this is not completely necessary. The trust management architecture presented in the next section will require trusted centralized components. However, the role of these components, and therefore, the performance penalty of using them, can be minimized. Thus, the achieved architecture is a compromise between the P2P and client-server models that is secure and benefits from increased scalability due to the distribution of most game activities.

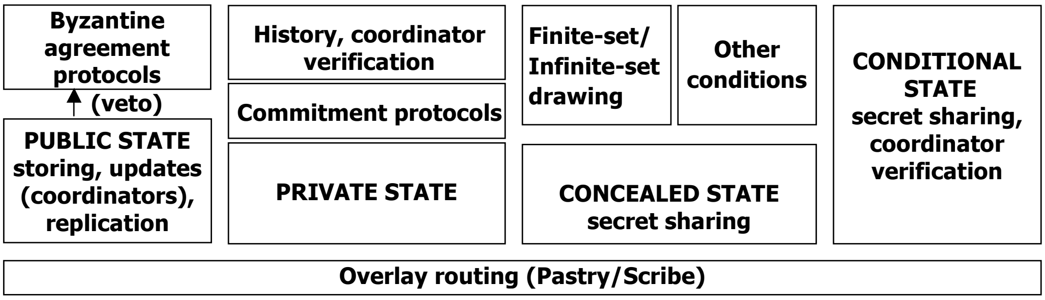

Figure 1: A trust management architecture for P2P MMO games.

3 TRUST ENFORCEMENT ARCHITECTURE

In this section, we propose a trust management architecture for P2P MMO games. Before the details of the proposed architecture will be described, let us shortly discuss the used concepts of "trust" and "trust management".

Trust enforcement

In much previous research, the notion of trust has been directly linked to reputation that can be seen as a measure of trust. However, some authors (Mui, L., 2003, Gmytrasiewicz, P., 1993) have already defined trust as something that is distinct from reputation. In this paper, trust is defined (extending the definition of Mui) as a subjective expectation of an agent about the behavior of another agent. This expectation relates the behavior of the other (trusted) agent to a set of normative rules of behavior, usually related to a notion of fairness or justice. In the context of electronic games, fair behavior is simply defined as behavior that obeys all rules of the game. In other words, an agent trusts another agent if the agent believes that the other agent will behave according to the rules of the game.

Trust management is used to enable trust. A trust management architecture, system or method enables agents to distinguish whether other agents can or cannot be trusted.

Reputation systems are a type of trust management architectures that assigns a computable measure of trust to any agent on the basis of the observed or reported history of that agent's behavior. Among many applications of this approach, the most prominent are on-line auctions (Allegro, E-Bay). However, P2P file sharing networks such as Kazaa, Mojo Nation, Freenet | Freedom Network also use reputation. Reputation systems have been widely researched in the context of multi-agent programming,, social networks, and evolutionary games (Aberer, K., 2001).

Our approach does not rely on reputation, which is usually vulnerable to first-time cheating. We have attempted to use cryptographic methods for verification of fair play, and have called this approach "trust enforcement". To further explain this approach to trust management, consider a simple "real-life" analogy. In many commercial activities (like clothes shopping) the actors use reputation (brand) for trust management. On the other hand, there exist real-life systems that require and use trust management, but do not use reputation. Consider car traffic as an example. Without trust in the fellow drivers, we would not be able to drive to work every day. However, we do not know the reputation of these drivers. The reason why we trust them is the existence of a mechanism (the police) that enforces penalties for traffic law violations (the instinct for self-preservation seems to be weak in some drivers). This mechanism does not operate permanently or ubiquitously, but rather irregularly and at random. However, it is (usually) sufficient to enable trust.

The trust management architecture proposed in this paper is visualized on Figure 1. It uses several cryptographic primitives such as commitment protocols and secret sharing. It also uses certain distributed computing algorithms, such as Byzantine agreement protocols. These primitives shall not be described in detail in this paper for lack of space. The reader is referred to (Menezes, J., 1996, Lamport, L., 1982, Tompa, M., 1993). The relationships between the components of the trust management architecture will be described in this section.

Our trust management architecture for P2P MMO games will use partitioning of game players into groups, like in the approach of (Knutsson, B., 2004). A group is a set of players who are in the same region. All of these players can interact with each other. However, players may join or depart from a group at any time. Each group must have a trusted coordinator, who is not a member of the group (he can be chosen among the players of another region or be provided by the game managers). The coordinator must be trusted because of the necessity of verifying private state modifications (see below). However, the purpose of the trust management architecture is to limit the role of the coordinator to a minimum. Thus, the performance gains from using the P2P model may still be achieved, without compromising security or decreasing trust.

Game play scenarios using trust management

Let us consider a few possible game play scenarios and describe how the proposed trust management mechanisms would operate. In the described game scenarios that are typical for most MMO games, the game state can be divided into four categories:

- Public state is all information that is publicly available to all players and such that its modifications by any player can be revealed.
- Private state is the state of a game player that cannot be revealed to other players, since this would violate the rules of the game.
- Conditional state is state that is hidden from all players, but may be revealed and modified if a condition is satisfied. The condition must be public (known to all players) and cannot depend on the private state of a player.
- Concealed state is like conditional state, only the condition of the state's access depends on the private state of a player.

Player joins a game. From the bootstrap server or from a set of peers, if threshold PKI is used, the player must receive an *ID* and a public key certificate $C=\{ID, K_{pub}, s_{join}\}$ (where s_{join} is the signature of the bootstrap server (or the peers), and K_{pub} is the public key that forms a pair with the secret key k_{priv},) that allows strong and efficient authentication (see next section. Note that the keys will not be used for data encryption). The player selects a game group and reports to its coordinator (who can be found using DHT routing). The coordinator receives the player's certificate. The player's initial private state (or the state with which he joins the game after a period of inactivity) is verified by the coordinator. The player receives a **verification certificate** (*VC*) that includes a date of validity and is signed by the coordinator.

Player verifies his private state. In a client-server game, the game server maintains all private state of a user, which is inefficient. In the P2P solution, each player can maintain his own private state, causing trust management problems. We have tried to balance between the two extremes. It is true that a trusted entity (the coordinator) must oversee modifications of the private state. However, it may do so only infrequently. Periodically or after special events, a player must report to the coordinator for verification of his private state. The coordinator receives the initial (recently published) private state values and a sequence of modifications that he may verify and apply on the known private state. For each modification, the player must present a proof. If the verification fails, the player does not receive a confirmation of success. If it succeeds, the player is issued a VC that has an extended date of validity and is signed by the coordinator. Verification by a coordinator is done by "replaying" the game of the user from the time of the last verification to the present. The proofs submitted by the player must include the states of all objects and players that he has interacted with during the period.

Player interacts with a public object. Peer-to-peer overlays (like DHTs) provide an effective infrastructure for routing and storing of public knowledge within the game group. Any public object of the game is managed by some peer. The player issues modification requests to the manager *M* of the public object. The player also issues a commitment of his action *A* that can be checked by the manager. Let us denote the commitment by $C(ID, A)$ (the commitment could be a hash function of some value, signed by the player).

Commitments should be issued whenever a player wishes to access any object, and for decisions that affect his private state. Commitments may also be used for random draws (Wierzbicki, A., 2004). It will be useful to regard commitments as modifications of public state that is maintained for each player by a peer that is selected using DHT routing, as for any public object.

The request includes the action that the player wishes to execute, and the player's validation certificate, *VC*. Without a valid certificate, the player should not be allowed to interact with the object. If the certificate is valid, and the player has issued a correct commitment of the action, the manager updates his state and broadcasts an update message. The manager also sends a signed testimony $T=\{t, A, S_i, S_{i+1}, P, s_M\}$ to the player. This message includes the time *t* and action *A*, state of the public object before (S_i) and after the modification (S_{i+1}) and some information *P* about the modifying player (f. ex., his location). The player should verify the signature s_M of the manager on the testimony. The manager of the object then sends an update of the object's state to the game group.

If any player (including the modifying player, if T is incorrect) rejects the update (issues a veto), the coordinator sends T to the protesting player, who may withdraw his veto. If the veto is upheld, a Byzantine agreement round is started. (This kind of Byzantine agreement is known as the Crusader's protocol.)

Note that if a game player has just modified the state of a public object and has not yet sent an update, he may receive another update that is incorrect, but will not veto this update, but send another update with a higher sequence number.

To decide whether an update of the public state is correct, players should use the basic physical laws of the game. For example, the players could check whether the modifying player has been close enough to the object. Players should also know whether the action could be carried out by the modifying player (for example, if the player cuts down a tree, he must possess an axe). This decision may require knowledge of the modifying player's private state. In such a case, the modification should be accepted if the modifying player will undergo validation of his private state and present a validation certificate that has been issued after the modification took place.

Player executes actions that involve randomness. For example, the player may search for food or hunt. The player uses a fair random drawing protocol (Wierzbicki, A., 2004) (usually, to obtain a random number). This involves the participation of a minimal number (for instance, at least 3) of other players that execute a secret sharing together with the drawing player. The drawing player chooses a random share l_0 and issues a commitment of his share $C(ID, l_0)$ to the manager of his commitments (that are treated as public state). The drawing player receives and keeps signed shares $l_1,...,l_n$ from the other players, and uses them to obtain a random number. The result of the drawing can be obtained from information that is part of constant game state (drawing tables).

Player meets and interacts with another player. For example, let two players fight. The two players should first check their validation certificates and refuse the interaction if the certificate of the other player is not valid. Before the interaction takes place, both players may carry out actions $A_1,...,A_k$ that modify their private state (like choosing the weapon they will use). The players must issue commitments of these actions. The commitments must also be sent to an arbiter, who can be any player. The arbiter will record the commitments and the revealed actions. After the interaction is completed, the arbiter will send both players a signed testimony about the interaction.

If the interaction involves randomness, the players draw a common random number using a fair drawing protocol (they both supply and reveal shares; shares may also be contributed by other players).

Finally, the players reveal their actions to each other and to the arbiter. The results of the interaction are also obtained from fixed game information and affect the private states of both players. The players must modify their private states fairly, otherwise they will fail verification in the future (this includes the case if a player dies. Player death is a special case. It is true that once a player is dead, he can continue to play until his VC expires. This can be corrected if the player who killed him informs the group about his death. Such a death message forces any player to undergo immediate verification if he wishes to prove that he is not dead). Note that at any time, both players are aware of the fair results of the interaction, so that a player who has won the fight may refuse further interactions with a player who decides to cheat.

Player executes an action that has a secret outcome. For example, the player opens a chest using a key. The chest's content is conditional state. The player will modify his private state after he finds out the chest's contents. To determine the outcome, the player will reconstruct conditional or concealed state.

Concealed state can be managed using secret sharing and commitment protocols, as described in (Wierzbicki, A., 2004). The protocol developed in (Wierzbicki, A., 2004) concerned drawing from a finite set, but can be extended to handle any public condition. The protocol has two phases: an initial phase and a reconstruction phase. The protocol required a trusted entity (in our case, the coordinator) that initializes concealed state by dividing the state into secret shares and distributing the shares to a fixed number of shareholders. The protocol also uses additional secret sharing for resilience to peer failures. Apart from the initializing of the state, the coordinator does not participate in its management.

The player issues a commitment of his action that is checked by the shareholders. If the condition is public but depends on the player's private state, the player decides himself whether the condition is fulfilled (he will have to prove the condition's correctness during verification in the future). If the

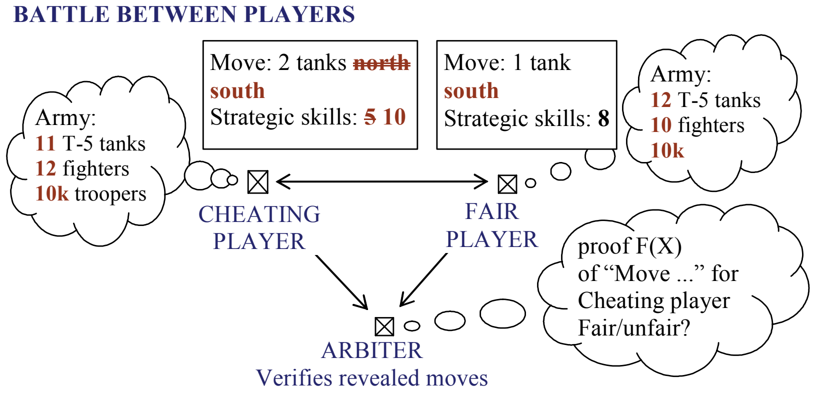

Figure 2: Preventing self-modification of private state.

condition to access an object is secret, the condition itself should be treated as a conditional public object. When the player has reconstructed the object, he must keep the shares for verification.

Note that concealed and conditional public objects can have states that are modified by players. If this is the case, then each state modification must be followed by the initial phase of the protocol for object management.

Authentication requirements

A P2P game could use many different forms of authentication. At present, most P2P applications use weak authentication based on nick names and IP addresses (or IDs that are derived from such information). However, it has been shown that such systems are vulnerable to the Sybil attack (Douceur, J., 2002).

Most of the mechanisms discussed in this paper would not work if the system would be compromised using the Sybil attack. An attacker that can control an arbitrary number of clones under different IDs could use these clones to cheat in a P2P game. The only way to prevent the Sybil attack is to use a strong form of authentication, such as based on public-key cryptography. Public key cryptography will be used in our trust management architecture for authentication and digital signatures of short messages, but not for encryption.

In a P2P game, authentication must be used efficiently. In other words, it should not be necessary to repeatedly authenticate peers. The use of authentication could depend on the game type. For instance, in a closed game, authentication could occur only before the start of the game. Once all players are authenticated, they could agree on a common secret (such as a group key) that will be used to identify game players, using a method such as the Secure Group Layer (SGL) (Agrawal, D., 2001). A solution that is well suited to the P2P model is the use of threshold cryptography for distributed PKI (Nguyen, H., 2005).

4 SECURITY ANALYSIS

In this section, the attacks illustrated in section 2 will be used to demonstrate how the proposed protocols protect the P2P MMO game.

Private state: Self-modification

Self-modification of private state can concern the parameters of a player, the player's secret decisions that affect other players, or results of random draws. The first type of modification is prevented by the need to undergo periodic verification of a private player's parameters. The verification is done by the coordinator on the basis of an audit trail of private state modification that must be managed by any player. Each modification requires proof signed by third parties (managers of other game objects, arbiters of player interactions). Any modification that is unaccounted for will be rejected by the coordinator. Players may verify that their partners are fair by checking a signature of the coordinator on the partner's private state. If a player tries to cheat during an interaction with another player by improving his parameters, he may succeed, but will not pass the subsequent verification and will be rejected by other players.

Modification of player's move decisions or results of random draws is prevented by the use of commitment protocols. The verification is made by an arbiter, who can be a randomly selected player (see Fig. 2). The verification is therefore subject to

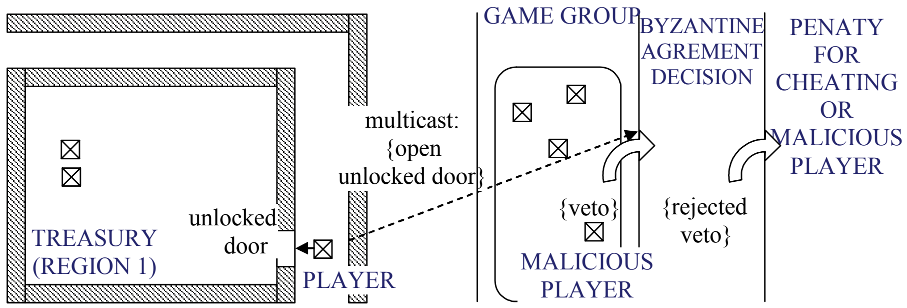

Figure 3: Byzantine agreement / veto protection of public state updates.

coalition attacks; on the other hand, making the coordinator responsible for this verification would unnecessarily increase his workload.

Note that in order for verification to succeed, the coordinator must possess the public key certificates of all players who have issued proof about the player's game. (If necessary, these certificates can be obtained from the bootstrap server). However, the players who have issued testimony need not be online during verification.

A player may try to cheat the verification mechanism by "forgetting" the interactions with objects that have adversely affected the player's state. This approach can be defeated in the following way. A player that wishes to access any object may be forced to issue a commitment in a similar manner as when a player makes a private decision. The commitment is checked by the manager of the object and must include the time and type of object. Since the commitment is made prior to receiving the object, the player cannot know that the object will harm him. The coordinator may check the commitments during the verification stage to determine whether the player has submitted information about all state changes.

Public state: Malicious modifications

We have suggested the use of Byzantine algorithms further supported by a veto mechanism (Crusader's protocol) to protect public state against illegal/malicious modifications. Any update request on the public state shall be multicast to the whole game group. The Byzantine verification within the group shall only take place when at least one of the players vetoes the update request of some other player. The cheating player as well as the player using the veto in unsubstantiated cases may be both penalized by the group by exclusion from the game (see Fig. 3). Such mechanism will act mostly as a preventive and deterring measure, introducing the performance penalty only on an occasional-basis.

The protection offered by Byzantine agreement algorithms has been discussed in (Lamport, L., 1982). It has been shown that the algorithms tolerate up to a third of cheating players (2N+1 honest players can tolerate N cheating players). Therefore, any illegal update on the public state will be excluded as long as the coalition of the players supporting the illegal activity does not exceed third of the game group. We believe such protection is far more secure than coordinator-based approach of Knutsson and tolerable in terms of performance. Performance could be improved if hierarchical Byzantine protocols would be used.

Attacks on P2P overlays

In our security architecture, players rarely reveal sensitive information. A player does not disclose his own private state, but only commitments of this state. Concealed or conditional state is not revealed until a player receives all shares. If the P2P overlay is operating correctly and authentication is used to prevent Sybil attacks, the P2P MMO game should be resistant to eavesdropping by nodes that route messages without resorting to strong encryption. A secure channel is needed during the verification of a players private state by the coordinator.

Concealed state: Attack on replication mechanism

Concealed state, as well as any public state in the game, must be replicated among the peers to be protected against loss. The solution of Knutsson uses the natural properties of the Pastry network to provide replication. However, we have questioned the use of this approach for concealed state, where the replicas cannot be stored by a random peer. The existence of concealed state has not been considered by Knutsson, and therefore they did not consider the fact that replicas may reveal the concealed information to unauthorized players.

In our approach for replication of concealed state, replication players are selected from outside the game group. This eliminates the benefits offered by Pastry network. On the other hand, this approach also eliminates the security risks. Please note that in our approach a certain number of players must participate to uncover specific concealed information. Therefore, a coalition with the replication player is not beneficial for a player within the game group.

5 PERFORMANCE ANALYSIS

We have tried to manage trust in a P2P MMO game without incurring a performance penalty that would question the use of the P2P model. However, some performance costs are associated with the proposed mechanisms. Our initial assumption about partitioning of game players into groups (sets of players who are in the same region) is required for good performance.

Byzantine protocols have a quadratic communication cost, when a player disagrees with the proposed decision. Therefore, their use in large game groups may be prohibitive. This problem may be solved by restricting the Byzantine agreement to a group of superpeers that maintain the public state (an approach already chosen by a few P2P applications, such as OceanStore). Another possibility is the use of hierarchical Byzantine protocols that allow the reduction of cost but require hierarchy maintenance.

Since private state is still managed by a player, it incurs no additional cost over the method of Knutsson. The additional cost is related to the verification of a player's private state by a coordinator. The coordinator must "replay" the game of a player, using provided information, and verifying the proofs (signatures) of other players, as well as the modifications of the verified private state. This process may be costly, but note that a coordinator need not "replay" all of the game, but only a part (chosen at random). This may keep the cost low, while still deterring players from self-modification of private state.

The cost of maintenance of concealed or conditional state is highest in the initialization phase (for a detailed analysis, see (Wierzbicki, A., 2005)). This stage should be carried out only when an object is renewed. During most game operations, the cost of concealed state management is reasonable. The reconstruction phase has a constant const (fetching the parts of an object). However, if the object's state changes, the object must be redistributed. The expense of this protocol may be controlled by reducing the constant number of object parts, at the cost of decreasing security. The number of object parts cannot be less than two.

All the proposed protocols have allowed us to realize one goal: limit the role of the central trusted component of the system (the coordinator). The coordinator does not have to maintain any state for the players. He participates in the game occasionally, during distribution of concealed/conditional state and during verification of private state. The maintenance of public state remains distributed, although it requires a higher communication overhead.

6 RELATED WORK

Several multi-player games (MiMaze, Age of Empires (Douceur, J., 2002)) have already been implemented using the P2P model. However, the scalability of such approaches is in question, as the game state is broadcasted between all players of the game. AMaze (Berglund, E., 1985) is an example of an improved P2P game design, where the game state is multicast only to nearby players. Still in both cases, only the issue of public state maintenance has been addressed. The questions how to deal with the private and public concealed states have not been answered (see section 4).

The authors of (Baughman, N., 2001) have proposed a method of private state maintenance that is similar to ours. They propose the use of commitments and of a trusted "observer", who verifies the game online or at the end of the game. However, the authors of (Baughman, N., 2001) have not considered the problem of concealed or conditional state. Therefore, their trust management architecture is incomplete. Also, the solution proposed in (Baughman, N., 2001) did not address games implemented in the P2P model.

The paper on P2P support for MMO games (Knutsson, B., 2004) offers an interesting perspective on implementing MMO games using the P2P model. The presented approach addresses mostly performance and availability issues, while leaving many security and trust issues open. In this paper, we discuss protocols that can be applied to considerably improve the design of Knutsson in terms of security and trust management.

7 CONCLUSION

The use of the peer-to-peer computing model has been restricted by problems of security and trust management for many applications. In this paper, we have attempted to show how a very sensitive application (a P2P Massive Multiplayer Online game) may be protected from unfair user behavior. We have been forced to abandon the pure peer-to-peer approach for a hybrid approach (or an approach with superpeers). However, we have attempted to minimize the role of the centralized trusted components.

The result is a system that, in our opinion, preserves much of the performance benefits of the P2P approach, as exemplified by the P2P platform for MMO games proposed by Knutsson. At the same time, it is much more secure than the basic P2P platform. The main drawback of the proposed approach is complexity. While we may pursue an implementation effort of the proposed protocols, a wide adoption of the peer-to-peer model will require a wide availability of development tools that include functions such as distributed PKI, efficient Byzantine agreement, secret sharing and reconstruction, and commitment protocols, that will facilitate construction of safe and fair P2P applications.

The approach that we have tried to use for trust management in peer-to-peer games is “trust enforcement”. It considerably different from previous work on trust management in P2P computing, that has usually relied on reputation. However, reputation systems are vulnerable to first time cheating, and are difficult to use in P2P computing because peers have to compute reputation on the basis of incomplete information (unless the reputation is maintained by superpeers). Instead, we have attempted to use cryptographic primitives to assure a detection of unfair behavior and to enable trust.

The mechanisms that form our trust management architecture work on a periodic or irregular basis (like periodic verification of private players by the coordinator or Byzantine agreement after a veto). Also, the possibility of cheating is not excluded, but rather the trust enforcement mechanisms aim to detect cheating and punish the cheating player by excluding him from the game. In some cases, cheating may still not be detected (if the verification, as proposed, is done on a random basis); however, we believe that the existence of trust enforcement mechanisms may be sufficient to deter players from cheating and to enable trust, like in the real world case of law enforcement.

REFERENCES

A.Wierzbicki, T.Kucharski: “P2P Scrabble. Can P2P games commence?”, Fourth Int. IEEE Conference on Peer-to-Peer Computing , Zurich, August 2004, pp. 100-107

A. Wierzbicki, T. Kucharski, Fair and Scalable P2P Games of Turns, Eleventh International Conference on Parallel and Distributed Systems (ICPADS'05), Fukuoka, Japan, pp. 250-256

B. Knutsson, Honghui Lu, Wei Xu, B. Hopkins, Peer-to-Peer Support for Massively Multiplayer Games, IEEE INFOCOM 2004

N. E. Baughman, B. Levine, Cheat-proof playout for centralized and distributed online games, INFOCOM 2001, pp 104-113

E.J. Berglund and D.R. Cheriton. Amaze: A multiplayer computer game. IEEE Software, 2(1), 1985.

J. Menezes, P. C. van Oorschot, S. A. Vanstone, Handbook of applied cryptography, CRC Press, ISBN: 0-8493-8523-7, October 1996

L. Lamport, R. Shostak, M. Pease, Byzantine Generals Problem, ACM Trans. on Programming Laguages and Systems, pp 382-401

M. Tompa and H. Woll, How to share a secret with cheaters, Research Report RC 11840, IBM Research Division, 1986

J. Douceur, The Sybil Attack, In Proc. of the IPTPS02 Workshop, Cambridge, MA (USA), 2002

D.Agrawal et al., An Integrated Solution for Secure Group Communication in Wide-Area Netwokrs, Proc. 6th IEEE Symposium on Comp. and Comm., 2001,

IETF, SPKI Working group

Zona Inc. Terazona: Zona application framework white paper, 2002

H. Nguyen, H. Morino, A Key Management Scheme for Mobile Ad Hoc Networks Based on Threshold Cryptography or Providing Fast Authentication and Low Signaling Load, T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 905 – 915, 2005

L. Mui, Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks, Ph.D. Dissertation, Massachusetts Institute of Technology, 2003

K.Aberer, Z.Despotovic (2001), Managing Trust in a Peer-To-Peer Information System, Proc. tenth int. conf. Information and knowledge management, 310-317

P. Gmytrasiewicz, E. Durfee, Toward a theory of honesty and trust among communicating autonomous agents Group Decision and Negotiation 1993. 2:237-258.