

"P2P Scrabble. Can P2P games commence?"

Adam Wierzbicki*

adamw@pjwstk.edu.pl

**Polish-Japanese Institute of Information Technology*

ul. Koszykowa 86, 02-008 Warsaw, Poland

Tomasz Kucharski*

Abstract

The article considers the design of P2P games without trusted, centralized resources. The main difficulty is how to prevent the possibility of cheating. The article considers Scrabble as a case study and attempts to solve issues such as maintenance of public, private, and concealed public state, as well as secret drawing from a finite set of objects. The issues of state replication are considered to allow node leaves. The article presents a fair protocol for secret drawing from a finite state that is resistant to node leaves.

Keywords

peer-to-peer, trust management, distributed hash tables, commitment protocols, secret sharing

1. Introduction

When considering the question of whether it is worthwhile to design P2P games, the same answers come to mind as for any other application: using the P2P model allows to avoid bottlenecks and single points of failure that occur in client-server applications. Additionally, P2P games would make better use of computational resources at the edge of the network. P2P games could also be designed to have smaller reaction times than client-server games.

However, the development of P2P games faces a significant obstacle: the issue of trust. In a client-server architecture, centralized management of the game state allows simple enforcement of the game rules. In a game without trusted, centralized resources, how can competing parties ensure fairness on their own?

This question may not be sufficiently specific to be considered directly. For that reason, we have tried to answer the question of avoiding cheating in a specific P2P game. For a case study, Scrabble – the favorite game of one of the authors – seemed a natural choice. This article describes our design of P2P Scrabble, and concludes by discussing the relevance of our results to the original question of general P2P game design.

2. Reputation-based mechanisms in P2P applications

The problem of trust management in a multi-agent system has been considered in many theoretical works and practical applications. One of the most common forms of trust management is the use of agent reputations. 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 [6,7,8] also use reputation. Reputation systems have been widely researched in the context of multi-agent programming. However, most of this research relied on a central point or reputation management that was assumed to be reliable and trustworthy. This assumption cannot be made in P2P systems, and therefore new research considers the use of reputation in P2P applications [2,3,4,16,17].

Reputation-based mechanisms could be used in games like P2P Scrabble. A player would receive a reputation based on a history of previous games, and this reputation could be used to exclude cheating players from a game. However, any reputation system has certain systematic drawbacks that have been a reason why it may be worth avoiding relying on these systems in P2P games. Among these, the most important is the problem of first-time cheating. Clearly, no reputation system can prevent any agent to build up a high reputation and then exploit it in order to cheat. (This is unfortunately understood by dishonest participants in some on-line auctions.) Another significant problem is the vulnerability of many reputation systems to coalitions of cheaters.

We believe that in many cases, cheating can be prevented in P2P applications without resorting to reputation-based systems. Reputations are necessary mostly if some functions of the system are influenced by circumstances that are not controlled by the system (such as the case of on-line auctions). Then, reputation-based mechanisms may be the only way to prevent cheating. In a P2P game such as P2P Scrabble, there may be no method to keep users from maliciously modifying information that is part of distributed storage. Since we

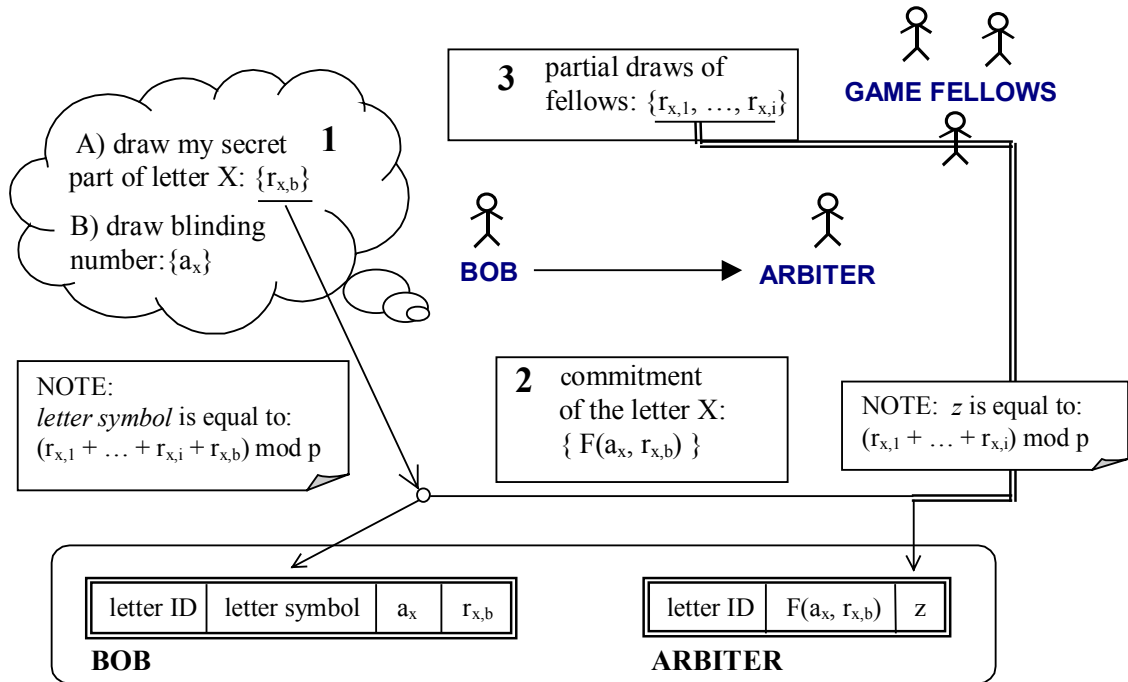


Figure 1 Random draws using commitment protocols and blinding

have no physical control over the computers of players, the best we can do is detect such behavior as quickly and efficiently as possible [18], and then exclude such users from the game or decrease their rating in a reputation-based system. This means that to some extent, we may use reputation systems in P2P games, but in this paper we shall attempt to discover better ways to prevent cheating.

3. Design of P2P Scrabble

Scrabble rules

All users of P2P Scrabble shall be called players. We shall refer to a Scrabble player who makes a move in the game as the drawing player. All other players that at this time point play the same game as the player shall be referred to as the competitors. All players that play a game together shall be referred to as the game group or the game players.

Scrabble is a game with turns, played usually by 3 or 4 players. Each player has a secret pool of letters that he tries to use to create words on the board. The players are awarded points for the words they put on the board, depending on their location on the board and the type of letters used. The letters are drawn from a letter sack. After each turn, every player must have a fixed number of letters (7). The correctness of words on the board is verified using dictionaries.

We shall not go into further detail of Scrabble rules, referring the reader to the game documentation.

However, some additional features of Scrabble will be explained further in the text.

Can we trust disinterested players?

The set of all players in P2P Scrabble is the set of all players currently playing all games. From the point of view of a game group, all other players are called *disinterested players*. Many problems with the design of P2P Scrabble could be solved if the game players could trust disinterested players.

However, this may not be as simple as it seems. There are two reasons why disinterested players cannot be wholly trusted: first, the players of a game could be in coalition with some disinterested players (in other words, these players may not be disinterested at all). Second, the disinterested players could be malicious: for the sake of spoiling the game for others (and improving their own ranking), a disinterested player could reveal or falsify information related to the players of a game.

For this reason, the sharing of information with disinterested players must be limited to a minimum.

The state of the game

The state of a P2P Scrabble can be divided into several kinds. The simplest is the public state: state available for everyone to read, and to modify (under certain conditions). More difficult to manage is private state: state available only for one player to read and modify. Other players must have some form of control over the private state – this will be the subject of next sections. The last type of state is concealed public state:

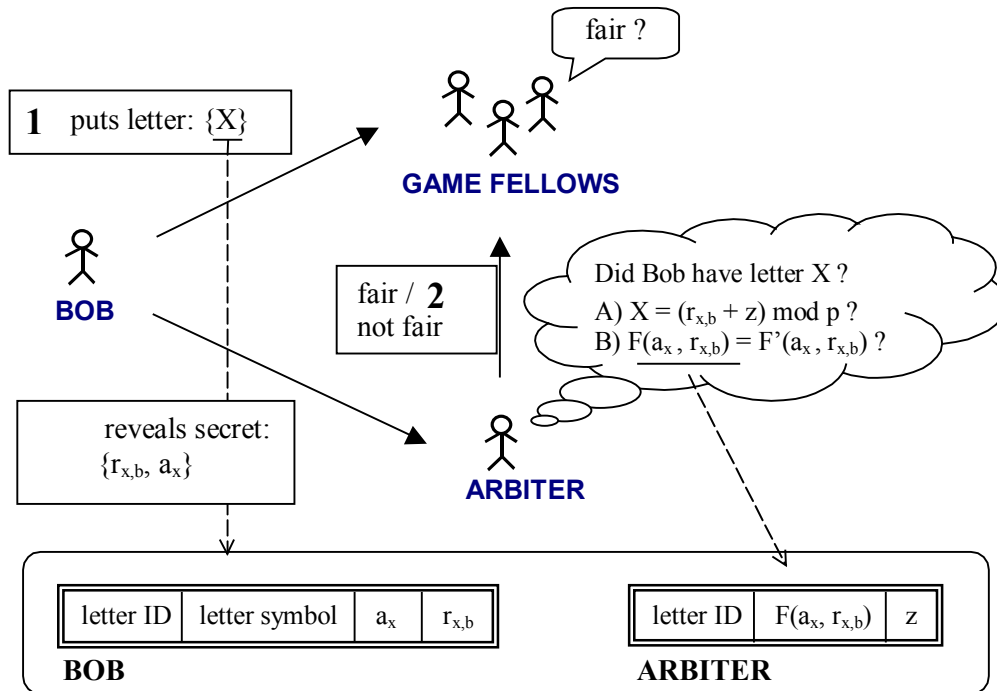


Figure 2 Verification of the draw using commitment

any game player can modify this state, but the game players cannot read it. This type of state will be discussed in the section "Management of the letter set". In this section, we shall discuss the simplest form of state in P2P Scrabble: public state.

The public state of P2P Scrabble consists of the board and the letters on the board, and of information about the player that has the turn. Additional information that is required to manage the two other types of state may also become a part of the public state.

In our implementation of P2P Scrabble, we have used Distributed Hash Tables (DHT) [9,10,11] for the management of public state (specifically, Pastry [10]). This mechanism must be supplemented by a lightweight, distributed transaction protocol that assures that operations on public state are fair. In order for this protocol to work, the game players must have a strong form of identity, obtained from a PKI certificate or using the Web-of-trust model. This identity may be concealed from other player using anonymizing techniques; nevertheless, it is required to avoid cheating using clones of the cheating player. Due to lack of space, the protocol for public state management shall not be discussed here. In this article, we focus on problems of providing fairness for random draws in a P2P game.

The public state must be replicated. Replication in mechanisms that use DHT has been discussed in the literature [9,10,11].

Random draws in P2P Scrabble

In order to prevent the possibility of cheating by the player, it must be possible to prove to the competitors that the player has legitimately drawn a letter that he wishes to put on the board. A straightforward solution to this problem would be to make all draws public. However, such an approach would make it possible for the competitors to cheat by using the information about the letters that have been drawn by the player. Such information could be exploited by the competitors to prevent the player from using his letters. Other approaches, that would rely on making the information about letters drawn by the player available only to selected players, would have a similar drawback since any other player could maliciously share this information with all competitors.

Therefore, the player must be able to prove to the competitors that he/she has legitimately drawn a letter without revealing what letter has been drawn. At a first glance, this seems impossible, until we discover the concept of commitment protocols.

Commitment protocols

Commitment protocols can be used in any game that involves making choices, for instance, consider simple "paper, rock, scissors". In this game, the knowledge of the choice of one participant would aid another. Also, when a player makes a choice, he must not be able to change his mind. Commitment protocols are used to bind players to their choices and keep them concealed till the

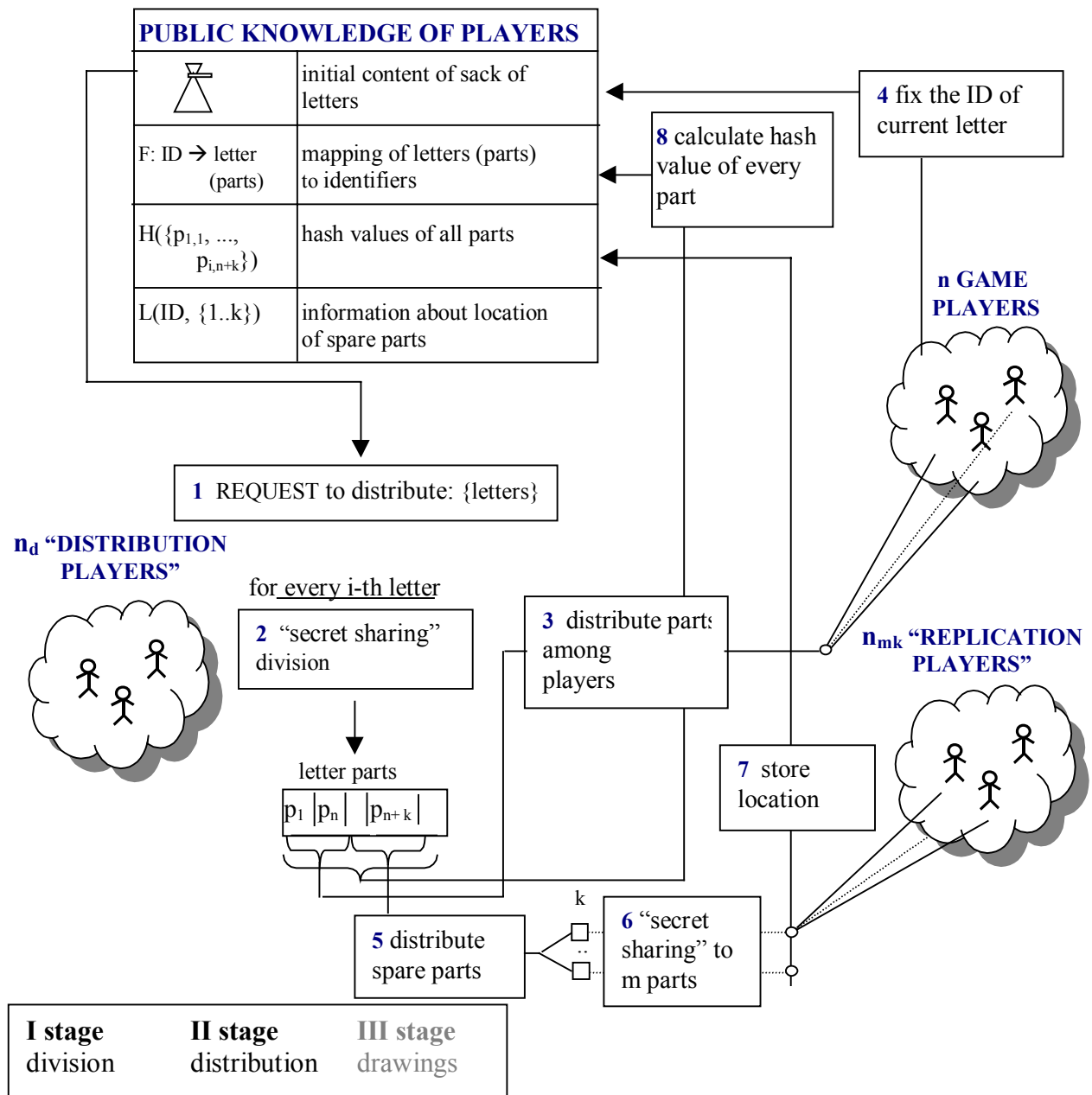


Figure 3 Letter division and distribution of parts

moment when all players are ready to verify their choices. How can this be done?

A commitment protocol must meet the following requirements in order to work:

- be able to produce a proof of the choice
- the proof on its own is not enough to reconstruct the choice (concealing the choice)
- the user is unlikely to be able to find two choices that produce the same proof (binding to choice)

The idea behind commitment protocols is to bind the choice using a strong mathematical function that a user cannot break in reasonable time (computational binding) or cannot break at all (information theoretical binding).

Among examples of such problems are: the reversal of a cryptographic hash function, or the discrete logarithm problem (DLP). Such a function can be used to produce a proof of the choice that can be made public without actually revealing information about the choice itself [9,13,14,15].

A function $F(x)$ that can produce a proof of a choice that would meet all the mentioned commitment protocol requirements must be close to monotonic and irreversible. A cryptographic hashing function has these properties, however, some additional modifications are needed to prevent a "replay attack". Details of how to construct this function are omitted for lack of space; see [12].

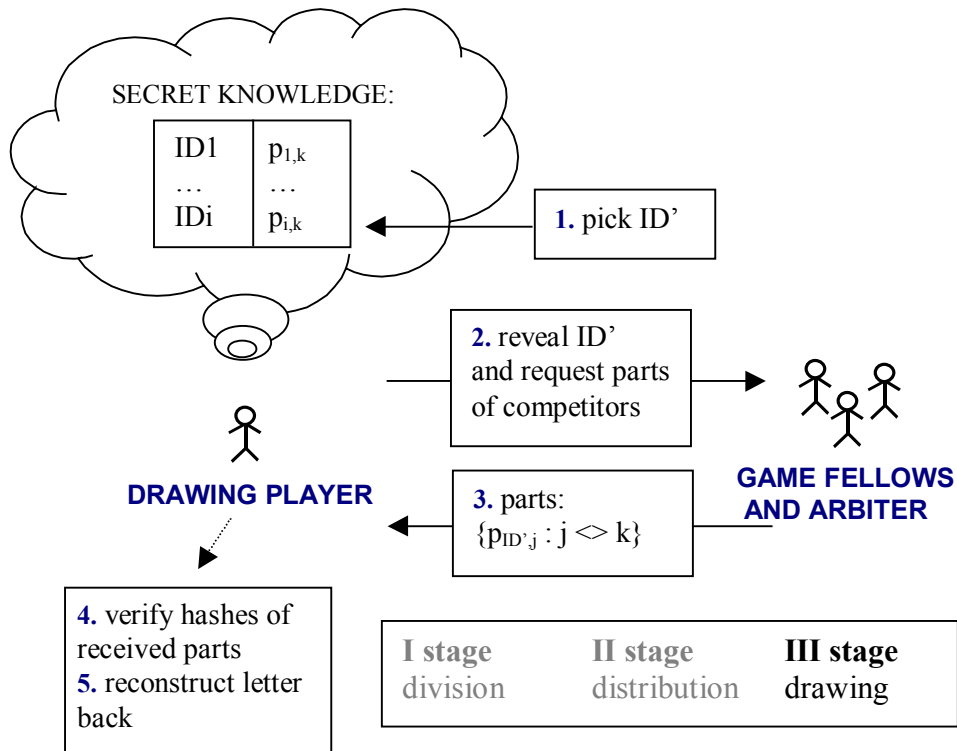


Figure 4 Drawing from a finite set

Introducing commitment schemes to P2P Scrabble

The presented commitment protocol allows a player to make choices on his own and later commit them. However, P2P Scrabble requires a more complex commitment protocol that would allow to commit the result of drawing which is not known by a player in advance (otherwise, a player would simply continue drawing letters until he found a letter that he likes). This sounds impossible – a player willing to commit the result of the drawing he still does not know and could not know before the commitment.

However, a simple trick, depicted in figure 2, might solve this problem. Bob wants to draw a new letter X and provide other users with the proof that he really has drawn this letter. The trick is to distribute the drawing process among many players. Each player draws a part, and later the combination of all parts produces a letter.

Bob has to draw his part first and keep it secret. This will be the only part missing to reconstruct the letter publicly. Bob commits his secret part using the presented commitment protocol, by sending it to an arbiter that can be any player. After that the competitors may draw their parts. Neither Bob nor the competitors can predict the final letter before their draws. Additionally, Bob is the only one that knows all the parts and may construct the final letter, while the competitors can not do so until the secret part is revealed. The commitment protocol should keep Bob from the

temptation of cheating by changing his secret part to change the final letter.

When Bob wishes to put one of his letters on the board, he must undergo a verification phase (see figure 3). Bob sends the letter and his committed, secret part to the arbiter and his game competitors. As all parts of the letter become public at that point, the game players can verify that the letter used by Bob really has been drawn by him. (step A in Figure 3). Additionally, the arbiter needs to verify that Bob's secret part has not been falsified (step B in Figure 3). Figure 3 presents an approach that additionally uses a blinding number in case the letter should be drawn again.

Management of the letter set

The presented drawing scheme has, however, serious limitations and without modifications, does not suffice for P2P Scrabble or a general P2P game. The algorithm was based on the assumption that the set of objects that can be drawn has an infinite number of elements of each type (or each element is returned to the set after the drawing). Referring back to P2P Scrabble – Bob could draw any letter no matter how many times it was drawn in the past. A draw of one player had no impact on following draws of other players. However, in P2P Scrabble there should be a finite set of letters – and Bob should only be able to draw a letter that still left in the set. This raises serious difficulties, as players should be able to draw objects from the set without revealing them

and at the same time other players should know that the object can no longer be drawn.

To understand the more complex drawing algorithm, one should first get acquainted with new problems that arise in drawing from a finite set. Later on the article will present one of the possible approaches to solve these problems. The main difference between infinite-set draws (i.e. presented in the previous section) and finite-set draws is that the latter ones imply the need to remember the state of the set of objects that can still be drawn. The following difficulties have to be faced in P2P Scrabble with finite-set drawing:

1) conceal content of the letter set

The knowledge of the content would allow players to easily figure out what the result of the secret drawing was. Players should be able to draw letters from the letter set without knowing which letters are still left within it.

2) replicate the letter set

P2P games have unpredictable nature. Nodes (players) may leave the game at any time, which requires algorithms that can still operate in such conditions.

3) prevent malicious modifications of the letter set

Players taking part in the game are distrustful, therefore none should ever rely on the other or group of others. Every player may attempt malicious action to modify the content of the letter set. In such a case, the player should be either prevented from such a possibility or it should become obvious to the competitors that the player is cheating.

Any solution to above difficulties must obviously distribute the letter set among many players, so that all of them have just partial knowledge of its content. Furthermore, the partial knowledge must be replicated by disinterested players in case any of the game players would leave the game. Finally, there must be some public one-way knowledge (consider hash functions) that would allow to verify that none of the game players modified its part of the letter set. The following list briefly summarize these requirements:

- no single game player should know the content of the letter set (distribution of knowledge)
- all game players are needed to draw a letter from the letter set (fairness)
- the drawing player is bound to the drawn letter and the letter is concealed from other players (binding & concealing)
- the content of the letter set is replicated among disinterested players
- public hash values are used to prevent malicious modifications

The suggested solution is based on the “secret sharing” model. Figures 4 and 5 present the algorithm

that satisfies all of the listed points. The actors taking part in the protocol are: n game players and $(n_{(mk)} + n_d)$ disinterested players (n_d distribution players, $n_{(mk)}$ replication players). Distribution players are involved in an initial distribution of the letter set among the game players (they shuffle the letters). Replication players are responsible for storing replicated parts of letters. The number of replication players - $n_{(mk)}$ - depends on the level of replication. The relation will be explained later in this section.

The algorithm is composed of three stages: division of letters, distribution of letter parts and drawing. The first two stages of the algorithm take place at the start of the game. (These stages might be used again whenever any player leaves the game, and his letter parts are restored from the spare parts.) To initiate the division and distribution process, a request with a set of letters should be passed to the distribution players. In the beginning of P2P Scrabble game, the set of letters is simply the initial content of the sack.

In the first stage, distribution players execute a secret sharing algorithm on each letter in the letter set. Each letter is divided into $(n+k)$ parts: $\{p_{i,1}, \dots, p_{i,n+k}\}$, where just n are enough to reconstruct the letter back. The remaining k parts could be considered as spare, replicated parts not needed as long as all the players are in the game. Secret sharing algorithms have been widely presented in the literature, therefore we shall not discuss details here (see: [12]).

Later on, in the second stage, the letter parts are distributed among players: n of them among n game players and k - among $n_{(mk)}$ replication players. The letter parts are sent by the distribution players to the game players in groups (letter by letter). The game players determine a unique ID for each received group of letter parts. The ID is used to identify n parts of a letter when it has to be reconstructed. Please note that a single player knows only his part $p_{i,j}$ of a letter and its ID. There is no way that the player might guess what the letter is knowing just its ID. In this way, game players may use IDs to refer to letters that are actually concealed from them. The only ones that know the whole letter, not just parts of it, are the distribution players. Therefore the mapping between letters and IDs must be kept secret from them. In the presented algorithm, encryption is a suggested solution to deliver the parts secretly to players. To prevent any malicious modifications of the letter parts, distribution players calculate hash values of every single letter part and make those public.

The replication of letter parts is more complex than it may seem at first. It is not enough to distribute k spare parts among k replication players. In such an approach, any player could easily find out the result of any drawing in a coalition with just one of the k replication players. (In the drawing stage all parts of the drawn letter become public except for the part of the drawing player. This

single unknown part could be simply filled with one of spare parts.) To minimize the risk of such coalitions, each of the k spare parts could be divided into further m parts using again a secret sharing algorithm. Then, a player would have to establish a coalition with m random disinterested players, which is very unlikely. Whenever any n_i players leave the game, n_i parts of any letter are lost, though spare parts can be used to reconstruct the remaining letters back. The recovered letters should be passed to distribution players to be divided once again, this time into $(n-n_i+k)$ parts, where just $(n-n_i)$ are enough to reconstruct a letter. These divided parts should be distributed among game players the same way as described in the second stage. Please note that k should be as large as the greatest number of players allowed to leave the game at once.

Let's move on to the third stage – the drawing. Please note again that every letter has its unique ID and its n parts are distributed among n game players. The drawing player picks one of the remaining IDs. To inform other players the letter is no longer in the letter set, the picked ID has to be revealed. The drawing player also asks his competitors for their parts, determined by the revealed ID. These $n-1$ parts become publicly known, the remaining one – kept by the drawing player – is a secret till the moment when the letter is used on the board. This single concealed part is enough to hide the drawn letter from other players. Note that the drawing player does not have to commit his secret part of the letter, since the hash values published by the distribution player prevent tampering with the secret part during reconstruction of the letter. The drawing player should also verify that those $n-1$ parts of the competitors were not modified in any way, by comparing hash values to the ones provided by distribution players.

To prove that the letter put on the board is the one a player has drawn, the player must reveal the ID and his secret part of the letter. At this moment, all of the letter parts are public. The competitors may reconstruct the letter to find out whether it is the one put on the board. The only thing left to verify is that none of the parts were actually modified at any point (hash values).

A rule of Scrabble that has not been so far considered is the possibility of returning all letters to the sack and drawing new ones at the cost of losing a turn. This can be implemented in a following manner. First, the drawing player broadcasts the IDs of the returned letters to the game group: these IDs become available for drawing. Second, all letters that are available for drawing are reconstructed and sent to the distribution players with a request to divide and distribute the letters once again among the game group. The reason for this step is that the drawing player now has some additional knowledge about the IDs of letters that he has returned. The new distribution reshuffles all remaining letters. Finally, the drawing player draws new letters.

4. Opportunities for cheating

Coalitions

The presented algorithm has one serious shortcoming. Coalitions between distribution players and game players are possible. The first ones know the letters, the latter ones know the IDs. The combination of this knowledge would obviously make all draws unfair. One could possibly get away with that using anonymous proxy communication between distribution players and game players. That way the coalition would have to include two random disinterested players – proxy and distribution player to make it worthwhile.

Another approach would be to assume that the game contains trusted supernodes that function as distribution players. Note that the role of the supernodes is limited to the beginning of the game and to dealing with node leaves (redistribution of letters).

Security of DHT

The public state of P2P Scrabble, and the replication of the letter set, require the use of resources of disinterested players. In our design of P2P Scrabble, we have used DHT as a mechanism of organizing a network of all players. However, we are aware of the potential security problems of DHT, that do not have satisfactory solutions in the available implementations of this mechanism [19]. Among possible approaches to improve DHT security for public state, we could mention [18].

To improve security of our implementation of P2P Scrabble we could: store the public state of a single game only at the nodes of the game players; limit the communications of a single game only to game players, replication players and distribution players; and use trusted supernodes as distribution players as suggested in the previous subsection. However, such an approach would require using a different mechanism that DHT for the organization of the game. Specifically, we would require a P2P platform that implements multicast, such as Rhubarb [5]. This is an issue for further work.

Safety of commitment protocols and secret sharing

The possibility of cryptanalytical attacks on the mechanisms described in this paper cannot be neglected. The security of the commitment protocols relies on the mathematical properties of the function F ; strong commitment protocols have been proposed in the literature that rely on computational or information-theoretical infeasibility of attacks. Secret sharing schemes exist that allow to discover cheating players (who reveal their share of the secret to other players). However, such subjects are beyond the scope of this paper.

Dictionary attacks

The limited vocabulary of letters and availability of hashes of letter parts makes dictionary attacks on the secret part feasible. Therefore, blinding techniques are required. One approach might involve blinding all letter parts of competitors before they are made public and then revealing the blinding numbers in the verifying stage.

5. Summary

The presented design of P2P Scrabble has been developed as a case study of a more general question: how to design P2P games without central coordination so that no player can cheat?

When considering this question in the light of our experience with P2P Scrabble, it is relevant to ask once again: why do we have to use a P2P model that strictly excludes central coordination? An addition of a central resource that is trusted by all players could simplify our design of P2P Scrabble.

The answer to this could be that now, when we have tried to design a game that does not use centralized resources and disallows cheating, we have learned how a small amount of centralized resources could be applied best to improve the design. For instance, replacing the distribution players with a single trusted supernode would certainly improve the design. Other such improvements are probably possible and could be the issue of future work.

Returning once again to the original question, we have to consider what other issues might arise in other P2P games that have not been considered in our case study. In other words, how can other games differ from Scrabble?

Several other games could use the design presented here for P2P Scrabble (consider, for instance, bridge or poker). However, all of these games are games of turns. In our design, we have not considered the issue of concurrency of drawing, assuming that the drawing player is the player who has the turn. Considering multi-user games without turns is an issue of future work.

Other questions that have not been considered in our design is how a player could modify the game state that influences the drawings of other users differently than in Scrabble. In Scrabble, the user only removes letters from the letter set: however, what if a user could add letters or replace letters in the set? We believe that our design could accommodate these changes, however, we decided to leave this issue for future work.

References

1. C.Farkas, G.Ziegler, A.Meretei, A.Lorincz (2002), Anonymity and Accountability in Self-Organizing

- Electronic Communities, Proc. ACM workshop Privacy in the Electronic Society, 81-90
2. K.Aberer, Z.Despotovic (2001), Managing Trust in a Peer-To-Peer Information System, Proc. tenth int. conf. Information and knowledge management, 310-317
3. M.Gupta, P.Judge, M.Ammar (2003), A Reputation System for Peer-to-Peer Networks, Proc. 13th int. workshop Network and op. sys. support for digital audio and video (ACM Press), 144-152
4. B.Yu, M.Singh (2002), An Evidential Model of Distributed Reputation Management, Proc. first int. joint conf. Autonomous agents and multiagent sys., part 1, 294-301
5. Wierzbicki, R. Strzelecki, D. Świerczewski, M. Znojek (2002), Rhubarb: a Tool for Developing Scalable and Secure Peer-to-Peer Applications, Second IEEE Int. Conf. Peer-to-Peer Computing, P2P2002,
6. Gnutella/ng, World Wide Web page, http://mangocats.com/annesark/gnutellang/wego_pages.html, 2000
7. Freenet, World Wide Web page, <http://freenetproject.org/cgi-bin/twiki/view/Main/WebHome>, 2002
8. Mojo Nation, World Wide Web page, <http://www.mojonation.net/>, 2000
9. Stoica, R. Morris, D. Krager, M. F. Kaashoek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications", Proceedings of ACM SIGCOMM'01 Conference, 2001
10. P. Druschel, A. Rowstron, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01), 2001
11. Zhao, J. Kubiawicz, A. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing", Technical Report CSD-01-1141, U.C.Berkeley, 2001
12. J. Menezes, P. C. van Oorschot, S. A. Vanstone, "Handbook of applied cryptography", CRC Press, ISBN: 0-8493-8523-7, October 1996
13. C.P. Schnorr, "Method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system", U.S. Patent # 4,995,082, 19 Feb 1991
14. M. Tompa, H. Woll, "Random self-reducibility and zero-knowledge interactive proofs of possession of information", Proc. IEEE 28th Annual Symposium on Foundations of Computer Science, 472482, 1987
15. J. Quisquater et al, "How to explain zero-knowledge protocols to your children", in G. Brassard, editor, Advances in Cryptology - CRYPTO '89, Lecture Notes in Computer Science, vol.435, pp.628-631, 1990
16. Singh, Ling Liu, "TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems", Proc. IEEE Peer-To-Peer Conference, 2003
17. Y. Wang, J. Vassileva, "Trust and Reputation Model in Peer-To-Peer Networks", Proc. IEEE Peer-To-Peer Conference, 2003
18. G. Caronni, M. Waldvogel, "Establishing Trust in Distributed Storage Providers", Proc. IEEE Peer-To-Peer Conference, 2003
19. E. Sit, R. Morris, "Security considerations for peer-to-peer distributed hash tables", Proc. IPTPS02 Workshop, 2002