Keeping Bots out of Online Games

Philippe Golle and Nicolas Ducheneaut
Palo Alto Research Center
3333 Coyote Hill Rd
Palo Alto, CA 94304 USA
{pgolle,nicolas}@parc.com

As multiplayer online gaming gains in economic and social importance, an increasingly large number of players is beginning to rely on "bots" (automated player agents) to gain unfair advantages in games. In this paper we study the problem of restricting participation in online games to human players, so they can enjoy the game without interference from these bots. We propose a range of techniques, both software and hardware based, to distinguish human players from bots. Our techniques are applicable in a wide variety of online games - from poker to "shoot'em ups." They are cost-effective, immune to cheating, and preserve the human players' enjoyment of each game. We conclude with a discussion of how game-theoretic approaches to *deter* the use of bots may complement our techniques to *detect* bots.

Categories and Subject Descriptors: H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities.

Additional Key Words and Phrases: Games, Agents, Bots, Reverse Turing Test, CAPTCHAs.

1. INTRODUCTION

Multiplayer computer games have become an increasingly important economic, social, and cultural phenomenon: nowadays millions of players routinely gather online to play their game of choice [Woodcock 2005]. Online game genres are extremely diverse: First-Person Shooters (FPS), Massively Multiplayer Online Role-Playing Games (MMORPGs), and online card games (e.g. poker) are among the most popular. However all share a similar characteristic: they pit *human* players against one another, unlike earlier forms of computer gaming where most opponents were computer-controlled. The inherent limitations of human physical and mental capabilities, the diverse ways in which these limitations manifest themselves in individuals, and the ability to transcend these limitations are all essential to the enjoyment of a multiplayer game. For instance, success at playing poker is largely predicated on one's ability to correctly assess probabilities; success in FPS depends in great part on hand-eye coordination and "twitch" reactivity; and success in MMORPGs depends on playing the game for long hours, collaborating with others to gain experience, and accumulating powerful objects.

Yet some players do not accept the rules of the game and refuse to play only to the best of their abilities. Instead, they use various forms of automation to gain an unfair advantage. The use of these so-called "bots" (or automated player agents) has become more prevalent recently because of the increasingly porous barriers between real and virtual economies [Castranova 2003]. For instance, writer Julian Dibell recently told Wired he earned $3,917 in one month by buying and reselling cybergoods from Ultima Online on eBay. The process of collecting artifacts and/or

in-game currency in MMORPGs can easily be automated – a practice referred to as "farming" that can lead to substantial monetary gains. In online card rooms, bots can be used to play games entirely based on their statistical properties, thereby earning money against imperfect human players [Brunker 2004]. Finally in FPS, bots can be used to increase a player's performance, for instance by artificially increasing targeting accuracy (CounterStrike's "aimbot" is one of the most blatant examples).

The problem of bot participation is not a superficial symptom of the limitations of existing games. Rather, it is an inherent consequence of the fact that repetition is a fundamental component of games [Koster 2004]. Mastering a "pattern" is an important source of enjoyment, and games with infinite variations are perceived as "chaotic" or "noisy" and poorly received by players. Thus bots cannot be eliminated only by designing richer, more diverse games (too much repetition attracts bots, too little drives humans away). Instead, techniques are needed to distinguish bots from human players so that bots can be eliminated from games.

The problem of distinguishing human players from bots in online games may not at first sound very difficult. After all, there is a clear gap between the abilities of humans and computers (e.g. bots can not carry coherent sustained conversations with humans). For our purposes however, identifying this gap is not enough. We must exploit it in games to distinguish bots from human players in a way that is *cost-effective, immune to cheating and that preserves the human players' enjoyment of the game*. These three requirements are not easily satisfied together and the best solution often depends on the game.

We propose two broad approaches to keep bots out of online games. The first consists of seamlessly integrating into online games software-based tests to tells humans and computers apart, known as Reverse Turing Tests or CAPTCHA tests [Ahn et al. 2003]. Our second contribution is to propose hardware instantiations of CAPTCHA tests. Hardware CAPTCHA tests are extremely versatile and suitable for use in a wide variety of online games, from card and board games to "shoot'em ups." Finally, we end with a discussion of how game-theoretic approaches to *deter* the use of bots may complement our techniques to *detect* bots.

2. MODEL

We assume throughout this paper that bots are undesirable, but a disclaimer is in order: not all uses of automation are malicious or detrimental to the quality of the game. In fact, the use of bots is sometimes an integral part of the games. For instance Second Life[1], a virtual world focused on content creation, provides extensive scripting capabilities so players can easily create new content. In other cases bots are tolerated when used to automate tedious tasks (such as, for example, macros in Star Wars Galaxies[2]). Bot use is generally forbidden when it spoils other players' enjoyment of the game (e.g. when fleecing poker opponents or instantaneously killing an opponent with an "aimbot") or when it is detrimental to the financial interests of game publishers (e.g. when selling game content on open markets).

[1] `http://www.secondlife.com`
[2] `http://www.starwarsgalaxies.com`

Naturally, our techniques should only be applied to eliminate problematic instances of bot uses. The decision of which bot uses are allowed and which are forbidden should be made by the game designers and/or the players. Our contribution is to offer tools to enforce these decisions. More precisely, our techniques help enforce one of two properties: human presence (the weaker property) or human play (the stronger property).

Human presence. A technique enforces "human presence" in a game if some input from a human is required to play the game. The requirement is only that a human contributes *some of* the interaction with the game, not necessarily all of it. In particular, the human may be assisted by a bot that controls the rest of the interaction with the game. In online poker for example, human presence means that a bot can only play under the supervision of a human. The cards may be chosen by the bot according to probabilities computed by the bot, but the bot cannot play without a human "baby-sitter". While human presence may seem like a weak property, it is actually surprisingly useful. Consider that the scope of bot involvement in online poker games, FPS or MMORPGs would presumably decrease considerably if every single bot had to be supervised by a human player.

Human play. Human play is a stronger property that requires that *all* interaction with the game software come from a human, without any involvement from a bot.

Adversarial model. We assume that the game software is tamper proof and that bots can only interact with the game via the input and output interfaces defined by the game. These assumptions are justified for online multiplayer games since the game software resides on secure game servers under the control of a game operator. Players may succeed in reverse-engineering or tampering with the code running on their local machines, but this local code is used only to exchange data with the central servers. Tampering with it does not free players from the constraints of interacting with the games via well-defined input and output interfaces.

We make the standard assumption that players are rational, and derive no utility from the *intrinsic* use of a bot (no defense appears possible against an adversary who has more intrinsic utility for using a bot than for winning the game).

3. CAPTCHA TESTS

There exists a variety of automated tests for telling humans and computers apart in an online environment. These tests are known collectively as CAPTCHA tests [Ahn et al. 2003], for "Completely Automated Public Turing Test to Tell Computers and Humans Apart." A CAPTCHA test is a program that can generate and grade tests that most humans can pass but that current computer programs cannot pass. The most widely used CAPTCHA tests rely on humans' ability to recognize randomly distorted text or images [Coates et al. 2001; Ahn et al. 2004]. Even the best computer programs fail these tasks with high probability.

CAPTCHA tests can be used during a game to verify *human presence*. A wide range of testing strategies is possible. A single CAPTCHA challenge may be presented when a player joins a game, or else repeated CAPTCHA challenges may be presented to the player at random intervals over the course of a game. The game may allow the player to sometimes not reply, or provide an incorrect answer. A

player that fails one or several challenges may be suspended from the game, either for a limited time or permanently (in which case all progress made in the game is lost). CAPTCHA tests deter bot participation most effectively in long stateful games where players have a lot to lose if they fail a test. Games played in short, stateless rounds may require more frequent tests.

We call this use of CAPTCHAs "out-of-band" because the CAPTCHA test is not integrated within the world of the game, but instead occurs outside of it. The main advantage of out-of-band CAPTCHAs is that they are extremely flexible and can be used with just about any game. However, out-of-band CAPTCHA tests suffer from the following limitations:

CAPTCHA tests are *disruptive*: they draw the player's attention away from the game. The interruption is only momentary (it takes only a few seconds for a human to solve a CAPTCHA) but breaks the suspension of disbelief that is so important to gaming. CAPTCHAs can also adversely affect the pace of a game: frequent requests to solve CAPTCHA tests would be intolerable to human players, since pacing and a sense of "flow" is essential to the enjoyment of a game [Koster 2004]. We address this issue in Section 3.1.

CAPTCHA tests can be *outsourced*: a CAPTCHA must be solved by a human, but that human need not be the person playing the game. For example, a bot playing a game without human assistance may forward the CAPTCHA tests it receives to low-cost workers that specialize in solving them. Games are particularly vulnerable to these so-called CAPTCHA relay attacks [Stubblebine and Oorschot 2004] because they make frequent repeated use of CAPTCHAs. The cost of setting up a relay attack may be prohibitive to solve a single CAPTCHA test (e.g. when opening a new email account), but it becomes negligible if it can be amortized over many tests, as in online games. Traditional digital CAPTCHAs may thus not be very effective to prevent bot participation in online games. We address this issue in Section 4.2.

3.1 Embedding CAPTCHAs Into Games

With a little design effort, CAPTCHA tests can be embedded into games in ways that are much less disruptive than out-of-band CATPCHAs. For most games, there is a trade-off between how well a CAPTCHA test distinguishes between a human and a bot, and how well it blends into the game environment. Correspondingly, game designers can choose from a spectrum of tests that range from most secure to least disruptive, depending on the severity of the threat posed by bots and the tolerance of players for changes to the game.

At one end of the spectrum, we can embed into games the same well-studied CAPTCHA tests that we proposed using in out-of-band fashion in the previous section. These tests have withstood a lot of scrutiny and are widely believed to be secure, but they are difficult to integrate perfectly seamlessly into games. The best we can hope for is to minimize the disruption, compared to the out-of-band use of CAPTCHAs. We illustrate this approach with the example of in-game resource collection or creation in MMORPGs. It would be consistent with the game's environment to require players to obtain "licenses" before allowing them to perform certain actions. Licences would take the form of an "exam" that is a CAPTCHA

in disguise. For example, a licence to craft items may require players to prove that they can "read engineering blueprints" (that is, solve a text-based CAPTCHA such as Gimpy [Ahn et al. 2004]); a license to hunt may require players to identify an animal in a picture (that is, solve an image-based CAPTCHA such as PIX [Ahn et al. 2004]), etc. Obtaining the license and passing the exam would be almost instantaneous and only adds a simple preliminary step to the regular sequence of actions required to collect or create resources.

At the other end of the spectrum, we find tests that are far less secure but can be integrated virtually seamlessly into games. These tests are CAPTCHAs only in a weak sense: they make it far harder to program bots that play the game, but not impossible (they will not stop a determined adversary). These tests consist of adding random variations to key aspects of the game. For example, the location of powerful items and resources in MMORPGs can be decided at run-time when a player chooses a particular mission or quest. This prevents "spawn camping" at hard-coded locations, which is highly susceptible to automation and the unusually quick accumulation of valuable game resources. Several recent games rely on this mechanism (e.g. Star Wars Galaxy).

Like out-of-band CAPTCHAs, embedded CAPTCHAs allow for a broad range of testing frequency and strategy. CAPTCHAs that are tightly integrated into, and consistent with, the game's environment do not negatively affect suspension of disbelief and are thus also less likely to be outsourced.

Applicability. Embedding CAPTCHAs into games is a suitable approach to ensure human presence in games that are slow-paced, have high-entropy environments and rules that are not too specific (most MMORPGs satisfy these requirements). A slow-paced game ensures that players have enough time to solve embedded CAPTCHAs. A high-entropy environment, such as a a rich graphical world, allow embedded CAPTCHAs to blend naturally into the environment. Finally, the absence of very specific rules makes it possible to embed CAPTCHAs that are not noticeably out of place.

3.2 The Game as CAPTCHA

Extending this approach further, the game itself may be regarded as a CAPTCHA. The test to distinguish bots from humans then becomes none other than the game. This approach is counter-intuitive: the very presence of bots in a game would seem to suggest that the game is a poor distinguisher between human players and bots, but that is not the case. The critical observation is that successful bots perform as well as humans along a single measure of success: the scoring mechanism of the game. A variety of other measures such as, for instance, the time spent in certain game locations or the patterns of activity may distinguish bots from human players. Game designers may rely on collaborative filtering or other models from artificial intelligence to learn how to compute clusters of bots and clusters of human players.

Inter-player conversations. We illustrate this approach with the example of inter-player conversations. Text-based or even audio conversation is naturally integrated in numerous multiplayer games - in fact, some MMORPGs are purposefully designed to maximize player-to-player interactions [Koster 2005]. Bots cannot engage in sustained conversation with human players, but they could engage in

meaningless conversation with other bots. Interaction metrics computed over the entire player population (see for instance [Ducheneaut and Moore 2004]) can separate clusters of bots from clusters of human players. The clustering algorithms used could be fixed or adapted dynamically according to real-time analysis of the conversation dynamics in each specific game. It should be clear from this description that, from the view-point of the game operator, the use of conversation to distinguish humans from bots is entirely automated and requires no human intervention. All the human intervention required is provided, freely and unwittingly, by other players.

A potential limitation of computing metrics over the whole user population to distinguish bots from human players is that the resulting models may produce a high number of false positives and false negatives. We propose two approaches to deal with expected high error rates:

Use models as filters. The first approach is to rely on clustering models only as a first filtering step: to improve accuracy, models may be combined with the CAPTCHA tests of Section 3.1: accounts suspected of being played by bots are served with an embedded CAPTCHA test.

Use models to adapt the game. Another (and perhaps complementary) approach is to rely on clustering models not to expel bots from the game, but to dynamically adapt the level of difficulty of the game to the performance of players. For instance, models can track a player's progress and make sure that, as soon as a task is mastered and it can be performed repeatedly, a different "higher level" task replaces it. This amounts to a "softer" approach to eliminating bots, while simultaneously preserving, and even enhancing the enjoyment of human players.

Limitations. If the models used to detect bot patterns came to be known, or if they could be reconstructed from observation, they could be defeated. The battle between bot designers and bot detectors might then take the flavor of the "arms' race" that pits virus writers against anti-virus software. More importantly perhaps, computing these metrics might be prohibitively expensive to some game producers since it requires a significant amount of data logging and mining.

4. PHYSICAL CAPTCHAS

To move beyond the limitations of the approaches we described above, we propose hardware-based "physical" CAPTCHA tests that distinguish human players from bots based on humans' ability to interact with the physical world. Human players can easily perform physical actions such as pressing a button, moving a joystick or tapping on a touch-sensitive screen, but it is difficult and expensive to design hardware that automates these physical tasks. Physical CAPTCHAs offer several advantages over the digital CAPTCHAs of the previous section: they are harder to outsource, they can be less intrusive and can offer the stronger property of *human play*.

A physical CAPTCHA is a device, such as a keyboard or a joystick, that accepts physical inputs (e.g. motion, pressure) and produces digital outputs. The property desired of a physical CAPTCHA device is that the only way to learn the digital output corresponding to a given input at a given time is to physically produce that

input at that time. To guarantee this property, a physical CAPTCHA device must be *tamper-proof* and *authenticate* its output.

Authentication. The output of the physical CAPTCHA must be authenticated, either to a remote game server or to the game software running on the user's PC or game console, to prove that it comes from the device at a given time.

Tamper-proof. A tamper-proof CAPTCHA device immediately loses its ability to authenticate outputs if an an attempt is made to tamper with it (e.g. the authentication key is wiped out from memory). This ensures that inputs can only be submitted to the physical CAPTCHA device via its physical input interface. Any attempt to bypass that interface (opening the device, rewiring its microcontrollers) causes the loss of the authentication key.

4.1 CAPTCHA Input Devices

One possible approach to implementing physical CAPTCHA is to turn existing game input devices into CAPTCHA devices. Joysticks and other small game input devices can be rendered tamper-proof at little cost, and adding authentication capability is also inexpensive. Larger devices, such as keyboards, would be comparatively more expensive to make tamper-proofs (we propose an alternative solution for keyboard-based games in the next section). We call a tamper-proof joystick that authenticates its communication with the game software a "CAPTCHA joystick". Games played with a CAPTCHA joystick ensure that there is no other way for a player to send inputs to the game than by physical action on the joystick (pressing, pushing, etc.) A human can exert this physical action much faster and much more reliably and easily than a machine.

CAPTCHA input devices ensure not only *human presence*, but also the stronger property of *human play*, since every interaction with the game software is mediated via the CAPTCHA input device. CAPTCHA input devices are well adapted to a wide variety of games and FPS in particular: they guarantee that "aimbots" and other artificial performance-enhancing techniques cannot be used. We believe that we will see more and more authenticated, tamper-proof CAPTCHA joysticks in the future: FPS games are progressively becoming akin to spectator sports, with regular tournaments offering large prize pools - which greatly increases the importance of ensuring that humans, not bots, are playing.

Limitations. CAPTCHA input devices combine two functionalities: 1) they record the player's input stream and relay it to the game software and 2) they ensure that the input stream comes from a human. While very well adapted to a number of games, this combination can also sometimes be unwieldy. For example, board or card games are more naturally played with a keyboard than with a joystick, but designing a tamper-proof, authenticated keyboard would be expensive and difficult, given the size of a keyboard. Consider also that there is a tremendous amount of "legacy" game hardware in use, and that users might balk at the cost of upgrading to a new CAPTCHA joystick. These examples show the need, in current circumstances, for separating the task of recording a player's input from the task of verifying that the player is human. We therefore propose a cheaper, more versatile alternative to CAPTCHA input devices: the CAPTCHA token.

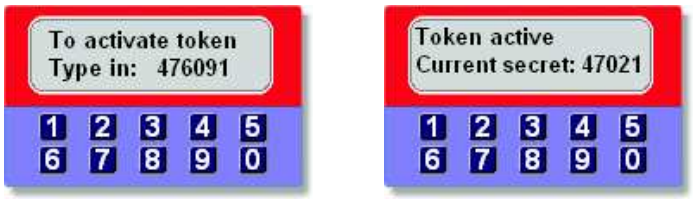

Fig. 1. A prototype CAPTCHA token (inactive state on the left; active on the right)

4.2 CAPTCHA Token

The only function of a "CAPTCHA token" is to test that a player is human. A CAPTCHA token does not serve as an input device to a game. CAPTCHA tokens are generic, versatile and cheap, and can be used in combination with just about any game. However, unlike CAPTCHA joysticks, they ensure only *human presence*, not *human play*.

CAPTCHA tokens can take different form factors. We describe an implementation that takes the form factor of a small calculator, equipped with a keypad and a screen (see Figure 1). The CAPTCHA token also has a clock (which need not be very accurate) and a CPU. We assume that the CAPTCHA token is tamper-proof, such that the only way to enter data is by physically depressing the keys on the keypad. We further assume that it would be difficult and expensive to design a mechanical machine that taps the keypad of the token in the right location. We estimate that CAPTCHA tokens would be inexpensive to manufacture in bulk (a few dollars each).

The token and the game server share a long (infinite) sequence of secret numbers $s_1, s_2, \ldots$ (each a few digits long). At time t (e.g. measured in minutes), the token authenticates itself to the game server with the value s_t. We assume that knowledge of the values $s_1, \ldots, s_t$ does not help an adversary learn anything about the value s_{t+1}. Standard cryptographic techniques allow such sequences to be generated and stored very efficiently: we may define $s_i = E_k(i)$ where E is a symmetric cipher (such as AES) and k a key known to the CAPTCHA token and the game server.

The CAPTCHA token can be in one of two states: active or inactive. When active, the token displays on its screen the value s_t corresponding to the current time t (expressed, say, in minutes). This value is updated every time the timer t is incremented. When inactive, the token does not display s_t but instead displays a random value c, which we call a challenge. To activate an inactive token, the user must type the challenge c on the keypad. If the correct value is entered, the token becomes active for a short period of time (say, one minute). After that, the token generates a new challenge c' at random, displays it on its screen, and automatically returns to the inactive mode. The new challenge must be typed to return the token to the active mode.

Use in Games. When a player signs up with an online game provider, the provider sends the player a CAPTCHA token. While playing the game, the player is occasionally asked for the current secret s_t. The player activates the token (which requires manually typing the challenge into the token's keypad), reads the value

s_t off the screen of the token, then sends this value to the game server (via the network connection of the game).

Comparison with other authentication tokens. The particular implementation of a CAPTCHA token that we have described bears a superficial resemblance to tokens used in two-factor authentication, such as RSA's pinpad [RSA 2005], but should not be confused with them. These two types of tokens offer completely different functionalities: tokens used for two-factor authentication let users prove knowledge of a secret, whereas CAPTCHA tokens let users prove that they are human (users need not remember any secret).

4.3 Properties of CAPTCHA Tokens

CAPTCHA tokens require an interruption on the part of the player and thus appear most suitable for slow-paced games with very specific rules and low-entropy inputs and outputs, such as card and board games (chess, poker, etc). These are precisely the games that are least amenable to the "CAPTCHA joystick" technique of Section 4.1. Thus CAPTCHA tokens and CAPTCHA joysticks appear to complement one another very well. In simulation/fantasy games such as current MMORPGs, the token could be used as part of the "licensing" process we described in Section 3.1, thereby preserving suspension of disbelief while offering even stronger guarantees that the player is human. Note also that since CAPTCHA tokens are hard to outsource (see below), verifications can be fairly infrequent (once a day or even once a week might suffice), which greatly reduces disruptions of the player's gaming experience.

Outsourcing. CAPTCHA tokens are difficult to outsource because they rely on a physical rather than digital challenge. For one thing, shipping a physical object requires a higher degree of trust than forwarding a digital challenge. Furthermore, it would be difficult to ensure consistent and reliable access to an outsourced token, since no redundancy is possible. The token occupies only one physical location, which may become unavailable for any number of reasons (the workers at that location may be asleep, or busy, etc.) Finally, it would be a logistical challenge to process many tokens in one location: storing hundreds or thousands of tokens in a way that allows for fast access (and without losing or misplacing any token) is hard. In summary, CAPTCHA tokens cannot be outsourced nearly as efficiently as digital CAPTCHAs that can be outsourced at virtually no cost to an anonymous, changing crowd of low-cost workers.

Automating the physical input. With the help of a webcam and optical character recognition (OCR) software, an adversary can read the current activation challenge c. Automatically typing the value c is harder: it requires customized hardware to tap the keys of the token in the right sequence. Attacks that require hardware are typically much harder to propagate than software attacks, since hardware cannot be downloaded but requires physical shipping.

If however hardware for automatically typing challenges on a token's keypad became sufficiently cheap and available, a simple counter-measure would be to replace the keypad of the token with a touch-sensitive area that could be overlayed on the token's screen (much like the screen of a PDA). Compared to a keypad, a

touch-sensitive screen offers a much broader method of data entry. For example, the token could draw a curve on the screen and ask the human to track that curve with a stylus. While humans can easily do so, automated hardware to do the same would be very costly.

5. DETERRING THE USE OF BOTS

In the previous sections, we have focused on techniques for *detecting* bots. Here, we discuss how the reward structure of games could be changed to *deter* the use of bots, following game theoretic principles. The distinction between detection and deterrence is fluid: the two techniques complement one another and should be used jointly. Our technique for deterring bots ensures that human players have no incentive to use bots because they stand to lose too much if caught using a bot.

The length of a game, and the degree to which it keeps states, influence how much of a punishment it is to be ejected from the game. Being ejected from a long-running game in which a player has a lot invested is a much more serious punishment that being ejected from a game played in short, stateless rounds. The following techniques help increase the amount that a player stands to lose if ejected from the game.

Increase the temporal scope of the game. Many bots are used to accumulate game resources quickly. These resources are later exchanged at a profit for in-game resources or currency, or for real-world currency through the mediation of open markets such as eBay. These exchanges are usually instantaneous if they take place within the game, or close to instantaneous if there is a transfer of real-world currency (auctions on eBay can end within hours). Therefore, even if bots are detected quickly, they can still be used to generate a significant volume of trade - after which a new game account can be created from scratch, a new bot is initialized, and the process is resumed. To prevent this, a delay could be introduced in transactions such that, for instance, it takes a few days to receive an in-game money transfer; similarly, it could take some time to "ship" (that is, transfer from one player account to the next) large volumes of resources. This makes using bots more problematic, as it greatly increases the time window the game has to detect bot use. This, in turn, increases the probability that the offending account will be cancelled - erasing the gains that were still to come from a money transfer or shipment. In turn-based games, a similar principle can be applied by grouping multiple rounds into "meta-rounds." Gains are then tallied and distributed only at the end of these meta-rounds. This creates a greater disincentive to be caught playing with a bot, since a player ejected from the game loses all the progress and investment made during a significant number of turns.

Increase the social scope of the game. Another alternative is to encourage players to team up with one another. For example, groups of friends may register together as a team for an online game. Alternately, players may team with other players they meet online. If one of the players belonging to a team misbehaves, punishment could be inflicted on the whole team. As an extreme example, the whole team may be excluded from the game if one player is caught using a bot. This ensures that social pressures keep all players honest. This form of self-policing, however, may be subverted by groups of players intent on victimizing a particular

individual [Bartle 2004], regardless of whether this individual is using bots or not. To minimize the risk of such "griefing" behavior [Foo and Koivisto 2004], players should team up preferably with "real-world" friends that they have known for a long time and trust.

6. CONCLUSION

The use of bots to assist or replace human players in multiplayer online games is becoming increasingly problematic. It is difficult to develop techniques to prevent bot use that are simultaneously cost-effective, immune to cheating, applicable to a wide variety of online game genres, and most importantly that preserve the human player's enjoyment of the game.

We argue in favor of hardware-based bot detection mechanisms: the CAPTCHA joystick and CAPTCHA token. The CAPTCHA token offers many advantages. First, it accurately separates bots from humans. Second, its physical nature offers the following benefits: it is cheap (based on commodity hardware), resistant to outsourcing, and its inputs cannot easily be faked. Its main drawback is that using the token can be mildly disruptive to the player, even though this is compensated by the fact that it only needs to be used infrequently. We believe that, as multiplayer gaming gains economic and social importance, hardware-based bot prevention techniques such as our CAPTCHA token will become increasingly prevalent. While we explored one approach in this paper, it is clear other hardware solutions should be designed and tested, especially if they can address the last remaining problem: disruption. We plan to explore ways to circumvent this problem in future work.

REFERENCES

Ahn, L. V., Blum, M., Hopper, N., and Langford, J. 2003. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt '03*. Springer Verlag, 294–311.

Ahn, L. V., Blum, M., and Langford, J. 2004. Telling humans and computers apart automatically. *Communications of the ACM 47,* 2 (Feb.), 57–60.

Bartle, R. 2004. *Designing virtual worlds*. New Riders Publishing, Indianapolis, IN.

Brunker, M. 2004. *Are poker 'bots' raking online pots? Some suspect computer programs have infiltrated Internet games*. http://www.msnbc.msn.com/id/6002298/.

Castranova, E. 2003. On virtual economies. *Game Studies 3,* 2 (Dec.).

Coates, A., Baird, H., and Fateman, R. 2001. Pessimal print: A reverse Turing test. In *Proceedings of the International Conference on Document Analysis and Recognition*. 1154–1159.

Ducheneaut, N. and Moore, R. 2004. The social side of gaming: a study of interaction patterns in a massively multiplayer online game. In *Proceedings of the ACM conference on Computer-Supported Cooperative Work*. ACM Press, 360–369.

Foo, C. and Koivisto, E. 2004. Grief player motivations. In *Proceedings of the Other Players conference*.

Koster, R. 2004. *A theory of fun for game design*. Paraglyph Press.

Koster, R. 2005. *Writings on game design*. http://www.legendmud.org/raph/gaming/.

RSA. 2005. *RSA SecurID pinpad*. http://www.rsasecurity.com/.

Stubblebine, S. and Oorschot, P. V. 2004. Addressing online dictionary attacks with login histories and humans-in-the-loop. In *Proceedings of the 2004 International Conference on Financial Cryptography*. Lecture Notes in Computer Science, vol. 3110. Springer Verlag, 39–53.

Woodcock, B. 2005. *An Analysis of MMOG Subscription Growth*. http://www.mmogchart.com/.