Spatial Game Signatures for Bot Detection in Social Games

Titus Barik[*] and **Brent Harrison**[*] and **David L. Roberts** and **Xuxian Jiang**
Department of Computer Science
North Carolina State University
{tbarik@, beharri5@, robertsd@csc. jiang@cs.}ncsu.edu

Abstract

Bot detection is an emerging problem in social games that requires different approaches from those used in massively multi-player online games (MMOGs). We focus on mouse selections as a key element of bot detection. We hypothesize that certain interface elements result in predictable differences in mouse selections, which we call *spatial game signatures*, and that those signatures can be used to model player interactions that are specific to the game mechanics and game interface. We performed a study in which users played a game representative of social games. We collected in-game actions, from which we empirically identified these signatures, and show that these signatures result in a viable approach to bot detection. We make three contributions. First, we introduce the idea of spatial game signatures. Second, we show that the assumption that mouse clicks are normally distributed about the center of buttons is not true for every interface element. Finally, we provide methodologies for using spatial game signatures for bot detection.

Introduction

While cheating is not a new concept in online games, the proliferation of "bots," or automated player agents that perform actions without any human intervention, has become a legitimate concern of game manufacturers (Yan and Randell 2005). Bot detection approaches have largely focused on massively-multiplayer online games (MMOGs), but there is an emerging problem with bots in social games. One class of attacks successfully utilized by bot authors in social games are human input device (HID) "mimicking attacks," where bots directly interact with the game by emulating mouse and keyboard actions through the game interface itself. Borrowing from biometrics, user behaviors including path movement, mouse selections, and to a degree the timing of interactions can potentially be used as an avenue for bot detection (Ahmed and Traore 2007). As a starting point for evaluating tractability in this domain, we focus on mouse selection events. Since the mechanics of every game vary, we hypothesize that interface differences induce mouse selection behaviors at the pixel level that are unique and specific to a particular game. Our main contribution in this paper is the introduction of a technique for identifying patterns in players' mouse behaviors, and changes in those patterns, that are unique to individual games and that can be used for bot detection. We call those patterns "*spatial game signatures*."

To evaluate this hypothesis, we deployed a word game (internally named Scrabblesque) that is representative in both the style and game mechanics of existing commercial social games. As participants played the game, data on their mouse click and mouse unclick actions were recorded. We performed an analysis of these actions to generate a set of spatial game signatures. We scored synthetically generated bot traces and known human player traces against these spatial game signatures to determine how effectively bot players can be differentiated from human players. The results of this study provide insight into the challenges for both bot detectors and bot authors when using mouse selection in a social game.

Although there are several existing HID approaches to bot detection for MMOGs, these detectors have an asymmetric advantage over bot authors in that these games have complex, highly-dynamic visual environments with rich user interactions (Gianvecchio et al. 2009). Social games have a simpler, mostly static visual environment, and a limited set of user interactions. Additionally, while MMOG bot detectors are traditionally desktop applications, social games are constrained to run in sandboxed, Web-browser environments, limiting their access to system resources. Bot detection approaches that use device drivers or system processes are not feasible for social games.

Social games require novel approaches outside of those used in MMOGs to successfully detect bots and disincentivize their usage. The spatial game signatures presented in this paper are a first step toward realizing that.

Related Work

Existing bot detection approaches can be classified along an axis that contrasts human interactive proofs (HIPs) against passive human observational proofs (HOPs).

HIPs, sometimes called reverse Turing tests, mandatory human participation, or CAPTCHAs, are an explicit means through which the system asks the user to perform a task that is relatively simple for a human player to solve, but difficult for a computer (Yampolskiy and Govindaraju 2008). While such approaches have been successfully used to hinder bots

[*] Both authors contributed equally to this work.

Figure 1: An implementation of Scrabble for collecting data.

in e-commerce and banking, online forums, free e-mail services, and online polls, HIPs present unique challenges in games. First, the HIPs cannot be a one-time challenge, otherwise the player could simply solve the HIP and then activate a bot. But more importantly, the use of multiple HIPs can be intrusive and interfere with the natural flow of the game (Golle and Ducheneaut 2005), making them difficult to implement in practice while still maintaining an immersive game experience for the player.

Efforts have been proposed to minimize the disruption of a player's game experience through HIPs in a way that is seamless within the game, such as by integrating mini-game challenges (Chow, Susilo, and Zhou 2010), or by adding random variations in key aspects the game, such as item locations. One limitation of these approaches is that authors must also create content that does not necessarily create a better experience for the player in order to satisfy an orthogonal security requirement. These difficulties suggest the need for passive, HOP approaches, such as spatial game signatures.

Data Collection

To test our hypothesis that spatial game signatures are a strong foundation to implement bot detection upon, we created a game based off of the popular game Scrabble. Scrabble is a word game where players receive letter tiles which they use to create words on a board. For the layout of the game, see Figure 1. The player's control buttons are shown across the bottom ("submit", "shuffle", "swap", and "pass"), and the set of tiles in their "rack" across the bottom are those they must use to create words on the main board (which covers the remaining portion of the interface). Study participants played against an AI opponent that used dictionary words to place tiles on the board. We chose to use Scrabble because it shares qualities with several social games in that there are only a few actions available to players and most of the interaction with the game is accomplished via rudimentary controls.

We collected the following analytics: mouse clicks and their associated position, mouse unclicks and their associated position, mouse position, changes in the player's rack, and the words that players have played. We have focused on mouse click and unclick data only in this paper. We discarded all clicks and unclicks that did not occur on buttons or on the game board. We normalized each click and unclick so that its pixel information was relative to the button it occurred in, rather than absolute pixel numbers.

To recruit participants for our study, we used snowball sampling through various online communities and social networks. We collected data from participants for a 4 week period. Upon clicking the link in our recruiting messages, participants were given consent information for participating in the study. Once they acknowledged the consent information, they were allowed to play the game as often as they liked. In total, we gathered data from more than 800 sessions, although only 514 of them resulted in valid data for our analysis. We used 70% (359 sessions) of these game sessions for training our bot detection algorithm and reserved the remaining 30% (155 sessions) for testing purposes. Sessions were assigned to training and test sets randomly.

Model Building and Scoring

Our approach consists of four phases: distribution fitting, candidate test generation, test selection, and score development.

Distribution Fitting

It is often assumed that mouse clicks are normally distributed about the center of a button. For certain arrangements of buttons, the standard deviation is constant for buttons that have the same area; however, these standard deviations must be empirically determined and are specific to each interface (Grossman and Balakrishnan 2005). While these results have been validated in controlled lab environments, these studies have not been validated in the wild. For this reason, our first focus was statistical normality tests to identify where input traces correlate (or not) with the existing literature on distributions.

Statistical normality tests assume that the input is continuous, but pixel-level user interactions are discrete. Because of this, and with the addition of typical human irregularity, these tests applied to the pixel-level frequency distributions of mouse selection events did not yield useful results. To identify what distribution fit the data best, we instead used the Akaike information criterion (AIC) with the maximum likelihood parameters for four different distributions. AIC determines the amount of information in the data that is lost when a selected distribution is used to describe it. We calculated AIC scores for Normal, Lognormal, Gamma, and Weibull distributions based on existing work (Bonto-Kane 2009).

AIC is calculated as follows:

$$\mathrm{AIC} = 2k - 2 \cdot \ln(L) \tag{1}$$

In the above equations, k refers to the number of free parameters the model has, and L refers to the log likelihood that a sample belongs to this model.

To determine the most likely fit, we used Akaike's rule of thumb. The distributions with the minimum and second lowest AIC scores are selected. If the score difference is less than 2, the distributions are indistinguishable by AIC, and *a priori* information should be used to model fit. For values

Table 1: Exact AIC scores for the shuffle button for four candidate distributions. Values in italics indicate the best fit distribution, based on minimizing information loss.

Name	X Axis		Y Axis	
	Click	Unclick	Click	Unclick
Normal	18344.1	18301.0	*16019.2*	*15988.2*
Weibull	*18317.0*	*18272.3*	16049.6	16018.8
Lognormal	18981.1	18914.8	16316.5	16283.0
Gamma	18653.8	18599.1	16126.8	16094.4

Table 2: Best fitting distribution identified by AIC. The four distributions considered were Normal, Lognormal, Weibull, and Gamma. Dashes indicate that differences in AIC scores for the two best distributions were less than 10.

Name	X Axis		Y Axis	
	Click	Unclick	Click	Unclick
Submit	Normal	Normal	Normal	Normal
Shuffle	Weibull	Weibull	Normal	Normal
Rack 1	Weibull	Weibull	Normal	Normal
Rack 5	Normal	Normal	Normal	Normal
Swap	-	-	-	-

between 3 and 10, the data may fit either depending on the sample. If the value is greater than 10, then there is a cogent argument of preferring the lowest scoring distribution over the second lowest (Ntzoufras 2009). An instance of this calculation is in Table 1. A summary of the best fit distributions for all relevant buttons can be found in Table 2. While most buttons are normally distributed, there are some exceptions, such as the shuffle button and the first rack button.

Candidate Tests

In order to characterize the identifiable changes in patterns of players' clicks we examined the data in aggregate. We examined the shape of the pixel-level distributions, the parameters associated with those distributions, and changes in those parameters that occur between buttons or modal interactions (*e.g.*, clicks vs. unclicks) on the same button.

We generated a set of candidate tests by running a pairwise paired-samples t-test on buttons of the same size, across both click and unclick conditions. For example, the submit and shuffle buttons have the same dimensions, and were compared for click and unclick distributions across the x and y dimensions independently. The results of these tests show that a vast majority of differences were significant ($p < 0.05$). An example of a resulting candidate test is: for the Rack 1 tile, the difference between the mean x position of the clicks (21.8) and unclicks (25.8) is significant. Table 3 contains a representative selection of the data that illustrates this. The complete set of pairwise t-tests forms a comprehensive test library.

Test Selection

We selected 10 tests that provide good coverage over the domain of spatial signatures. Table 3 contains a sample of exact

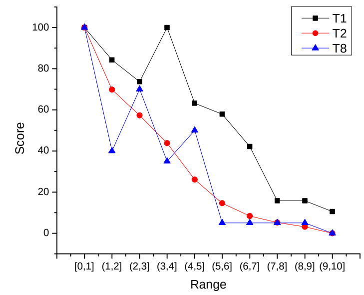

Figure 2: A sample of frequency distribution of scores for tests t_1, t_2, and t_8. Lower scores indicate behavior less similar to human behavior.

dimensions and pixel distribution parameter values. The specific values for an individual player are compared against the aggregate values to derive a "score" (which we will discuss in the next section). The tests are:

$\mathbf{t_1}$ The shuffle button has a Weibull distribution, as determined by AIC (see Table 2), in the x direction for both click and unclick actions with mean 29.1—to the right of the normalized center (25).

$\mathbf{t_2}$ The submit button has a similar mean for both clicks and unclicks in the x dimension, and this mean is slightly off center (26.6).

$\mathbf{t_3}$ The same as t_2, but performed in the y dimension.

$\mathbf{t_4}$ The submit button standard deviation in the x dimension for clicks is 7.9.

$\mathbf{t_5}$ The submit button standard deviation for the y dimension for clicks is 6.0.

$\mathbf{t_6}$ The Rack 5 button standard deviation for the x dimension for clicks is 6.1.

$\mathbf{t_7}$ The Rack 5 button standard deviation for the y dimension for clicks is 5.4.

$\mathbf{t_8}$ A difference should exist between the mean x positions in click (21.8) versus unclick (25.8) for Rack 1.

$\mathbf{t_9}$ A difference should exist between the mean y positions in click (18.4) versus unclick (14.4) for Rack 1.

$\mathbf{t_{10}}$ The Anderson-Darling test for normality at the $p < 0.03$ significance level for the submit button in both the x and y dimensions for click and unclick events is used. The p value is decreased from the typical $p < 0.05$ to reduce false positives due to inherent human variation.

Scoring Method

To calculate scores for individuals, we examine how likely we are to observe the differences between the parameters that characterize the distributions of their mouse clicks/unclicks from the parameters that characterize the distributions of all other players in aggregate. To do so, for each test, we computed a frequency distribution using leave-1-out sampling. For each user in the training set of 359 players, we

Table 3: Click and unclick data, including button dimensions, total number of clicks and unclicks, percentage of users that performed the action, and the mean normalized x and y click and unclick positions with standard deviations.

		Count		Users (%)		Mean X Position		Mean Y Position	
Name	WxH	Click	Unclick	Click	Unclick	Click	Unclick	Click	Unclick
Submit	50x42	8663	8650	98.6	98.6	26.6 ± 7.9	26.7 ± 7.8	20.5 ± 6.0	20.5 ± 6.0
Shuffle	50x42	2601	2596	48.2	47.9	29.1 ± 8.2	29.1 ± 8.2	21.7 ± 5.3	21.7 ± 5.3
Rack 1	38x38	3361	1119	92.8	63.5	21.8 ± 6.3	25.8 ± 6.6	18.4 ± 5.4	14.4 ± 5.6
Rack 5	38x38	4134	1299	95.0	68.5	19.9 ± 6.1	23.1 ± 8.4	18.6 ± 5.4	15.0 ± 6.1

calculated the differences between their distribution parameters and the remaining 358 players' distribution parameters in aggregate. For each test, we created a set of bins, which span between 0 and 10. We discarded data for all users with a difference greater than 10. The frequency distributions were then normalized by dividing the value of each bin by the value of the largest bin and then multiplying by 100 so that all scores fall between 0 and 100. So, the pixel difference from test t_k for player i based on distribution parameter p is given by

$$t_k(i) = \left| p_i - \frac{\sum_{j \neq i} p_j}{n-1} \right| \tag{2}$$

where n is the number of players in the training set. Each test is then binned according to $\lceil t_k(i) \rceil$ to construct a frequency distribution over difference values, after which a player x, not from the training set, can then obtain a score s over all tests k by first computing

$$t'_k(x) = \left| p_x - \frac{\sum_j p_j}{n} \right| \tag{3}$$

for all players j in the training set, where n is the number of the players in the training set. The final score for player x can then be obtained as

$$s(x) = \frac{\sum_k w_k \cdot b_k(t'_k(x))}{\sum_t w_k} \tag{4}$$

where b_k is the number of players in bin $\lceil t'_k(x) \rceil$ divided by the number of players in the largest-size bin. Example bin frequency distributions are presented in Figure 2. With the understanding that some tests may be easier to evade than others, a tunable parameter w_k exists for each test t_k so that administrators can adjust the test weights based on their own observations of existing bots. In this paper, the weights are largely for illustrative purposes and have been selected such that they have negligible effect on the overall scoring performance (see Table 5).

Many tests have minimum click requirements because the functions that the tests rely on have underlying constraints. As an example, standard deviation requires a minimum of two clicks. Users are assigned a score of zero when they do not satisfy the pre-requisites for a given test to prevent bots from ignoring difficult tests to obtain a high score.

The above technique is used for every test except for t_{10}, the normality test. For this test, a user is given a score of 25% for each dimension (across clicks and unclicks) where the Anderson-Darling test cannot determine that the user's distribution is statistically different from normal.

Additionally a quality (Q-score) was also computed for each user. With the exception of the normality test (t_{10}), the quality is simply the number of actions the user performed for the tested parameter. Quality is a useful estimate for determining the accuracy of the score. A low quality implies that very few actions were used in the computation of the score, indicating that the score may be unreliable for bot detection. For tests that compare multiple parameters (such as clicks vs. unclicks), the quality is the aggregate of both actions. For instance, the Q-score for a player in test t_2 with 100 clicks and 100 unclicks would be 200.

Evaluation

We characterize how effective our game signatures are at differentiating bots from human players by comparing five simulated bots against player data reserved for testing.

Bot Design

We designed five bots that generate mouse click and unclick actions over the buttons. We believe that the behavior of four of the bots is characteristic of bots that are in use today. The fifth bot is more of a sanity check—it is impossible to detect by design, but not feasible to implement in practice.

1, 2. Center Bot (Ceiling/Floor) always clicks the center of a button, and uses either a mathematical ceiling or floor function when the center falls between two pixels in either the x or y coordinates. We expected that these Center Bots would pass centrality tests, but with a standard deviation of zero would fail standard deviation tests.

3. Uniform Bot randomizes click and unclick locations using a uniform distribution. It has a similar mean to Center Bots, and therefore may do well in centrality tests. Unlike Center Bots, however, it has non-zero standard deviation so may do better in those tests. We still expect it to do poorly overall due to its one-size-fits-all solution.

4. Normal Bot generates clicks and unclicks using a normal distribution around the center of the button with $\sigma = button_{width}/2$ for the x dimension and $\sigma = button_{height}/4$ for the y dimension. This bot will probably perform better than Uniform Bots because it will also pass uniformity tests (t_{10}).

5. Mimic Bot is constructed to obtain scores comparable to human players by using the empirical means and standard deviations of all of the buttons. Such a bot could be developed if the bot author gains access to the game signatures, or if they perform a replication study to collect these parameters on their own.

Table 4: Total S- and Q-scores for bots for a game session.

Type	S-Score (%)	Q-Score
Center Bot (Floor)	19.9	1700
Center Bot (Ceiling)	22.6	1700
Uniform Bot	24.0	1700
Normal Bot	37.8	1700
Mimic Bot	96.3	1700

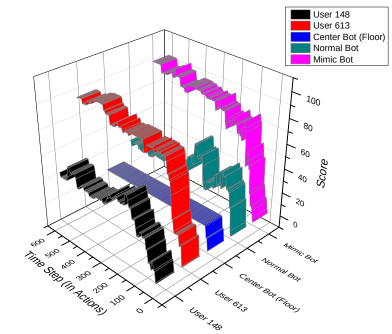

Figure 3: A comparison of the S-scores of two high-quality users against three of the bots.

Bot Evaluation

Table 4 is a summary of the scores for the bots. As expected, the Mimic Bot's score is comparable to that of high quality human players, given that it has *a priori* knowledge of the means and standard deviations for the buttons. It doesn't have a perfect score due to randomness in generation.

Table 5 shows the expanded view of the score calculations for all bots. For example, as hypothesized, the Center Bots and Uniform Bots do poorly in standard deviation tests (t_4, t_5, t_6, and t_7). Also, neither of these bots are able to successfully pass the normality tests (t_{10}), though the uniform distribution succeeds in a single dimension due to pseudo-randomness (S-score of 25). Further, as hypothesized, the Normal Bot passed the normality tests (t_{10}) with flying colors, but still scored poorly overall due to standard deviations different than the empirical values used in tests t_4, t_5, t_6, and t_7. Finally, the Normal Bot with a constant mean value for all buttons was unable to score well in tests comparing differences in means across different actions (t_8 and t_9).

Figure 3 illustrates the scores over time. For each data series, both the Q-score and S-score are recalculated after every five mouse actions. For human players (such as User 613), their scores rise relatively quickly as additional actions are performed. A minority of human players, such as User 148, obtain a low score despite their high quality simply because they deviate significantly from the aggregate training set. For Center Bots, there is an initial rise in score before the bot reaches steady-state. The Uniform Bot behaves similarly to the Center Bot, so it has been omitted from the figure. A steady-state property also exists for the Normal Bot, though its steady-state plateaus at a higher level because its distribution more closely matches that of human players.

One concern may be that high quality users may obtain

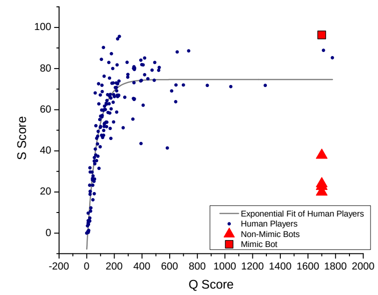

Figure 4: S-scores as a function of Q-scores for human and bot players. In general, a human's S-score will increase as they play longer whereas a bot's will not.

low scores, thus incorrectly classifying them as bots. The data suggests that for human players the S-score will generally increase exponentially as a function of Q-score, and this is confirmed by applying an iterative, reduced chi-square fit of $y = y_0 + Ae^{R_0 x}$ with $y_0 = 74.65 \pm 1.56, A = -82.43 \pm 3.17, R_0 = -0.013 \pm 0.001$ to obtain an adjusted R-square correlation of 83%. Indeed, as shown in Figure 4, the more actions a player performs, the higher their S-score, and the more likely that player is to be identified as a human player. The data in Figure 4 also indicates that for four of the five bots, despite having high Q-scores, their S-scores remain relatively low. The only bot with an S-score and Q-score in a region close to human players was the mimic bot which was designed to do so. There are two things to note. First, having access to the tests a bot detection engineer uses requires data only known by game designers, making implementation of a mimic bot in the real world difficult. Second, the large separation between the human players and bots in Figure 4 indicates that constructing a decision boundary for bot detection should be relatively straightforward.

Future Work

We have presented a method for creating spatial game signatures to aid in developing bot detection schemes. This paper focused on spatial game signatures using mouse selection events, although other mouse dynamics were recorded. We will use this data in the future to create additional signature types, especially with regard to mouse trajectories and timing. Many of the techniques that were used to score spatial game signatures could also be used to score temporal game signatures, or even spatiotemporal game signatures.

We chose a turn-based game as a baseline for bot detection approaches in social games. We would like to investigate this bot detection technique in more complex games. To strengthen the evaluation, we would also like to find actual bots that we can attempt to classify as human or not without prior knowledge in order to construct performance metrics.

One avenue for investigation is leveraging game signatures to provide the assurances that HIP techniques provide, while retaining the unobtrusiveness of HOP techniques. Imagine a player's Q-score is moderate and S-score is rel-

Table 5: Scores for all 10 tests for a selection of users from different score ranges and for all five bots. Higher S-scores indicate higher agreement with aggregate user behavior. Q-scores provide an estimate of the confidence of the score. Total S-scores are obtained by computing a weighted average of all of the tests. The weight of each test is indicated in parentheses.

		t_1 (1)	t_2 (1)	t_3 (1)	t_4 (3)	t_5 (3)	t_6 (2)	t_7 (2)	t_8 (2)	t_9 (2)	t_{10} (1)	Total
User 613	S-Score	15.8	100	82.8	100	100	100	100	100	75	50	88.8
	Q-Score	2	310	310	310	310	74	74	84	84	155	1713
User 148	S-Score	10.5	100	26.9	36.5	83.1	28.3	26.0	40	30	0	41.4
	Q-Score	27	109	109	109	109	11	11	22	22	54	583
User 119	S-Score	0	57.3	7.5	4.0	47.7	0	0	0	0	0	12.2
	Q-Score	0	6	6	6	6	1	1	1	1	3	31
Mimic Bot	S-Score	100	100	82.8	100	100	100	100	100	75	100	96.3
	Q-Score	200	200	200	200	200	100	100	200	200	100	1700
Normal Bot	S-Score	63.2	57.3	82.8	15.9	6.9	28.3	26.0	70	30	100	37.8
	Q-Score	200	200	200	200	200	100	100	200	200	100	1700
Uniform Bot	S-Score	100	69.8	100	0.8	0.8	1.1	0	35	30	25	24.0
	Q-Score	200	200	200	200	200	100	100	200	200	100	1700
Center (f) Bot	S-Score	63.2	69.8	100	1.6	0	0	0	50	10	0	19.9
	Q-Score	200	200	200	200	200	100	100	200	200	100	1700
Center (c) Bot	S-Score	100	100	82.8	1.6	0	0	0	50	10	0	22.6
	Q-Score	200	200	200	200	200	100	100	200	200	100	1700

atively low, putting them in a region of uncertainty. Using the game's mechanics to induce different interactions with the interface, we could increase their Q-score to make bot identification confidence higher. We might give the player four Z tiles, forcing them to use the "swap" button. In doing so, we would observe their click behavior in swap mode, and therefore increase their Q-score.

Conclusions

We have shown that spatial game signatures are a viable tool for bot detection. These game signatures are useful in detecting bots because they rely on empirical properties that are unique to the coupling between the game mechanics and the game interface. Further, these game signatures can reasonably be assumed to be unknown to bot authors.

For bots to successfully evade detection via spatial game signatures, it is not enough to use generic distributions. They must be aware of the game specific interactions that are unique to every game and cause changes in those generic distributions in certain situations. To discover these signatures, a bot author must either obtain these signatures by compromising the system in some manner, or by performing their own replication study to gather empirical data. Even if the bot author performs a replication study, they must implement the entire domain of spatial signatures, since they will be unaware of which subset of signatures are actually being used by the game for bot detection. If the game designers ever change the interface it will invalidate the bot's parameters. Therefore, we are enthusiastic that spatial game signatures will be a valuable tool for game designers to implement effective bot detection schemes in social games.

References

Ahmed, A. A., and Traore, I. 2007. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing* 4(3):165–179.

Bonto-Kane, M. V. 2009. *Statistical modeling of human response times for task modeling in HCI*. Ph.D. Dissertation, NC State University.

Chow, Y.-W.; Susilo, W.; and Zhou, H.-Y. 2010. CAPTCHA challenges for massively multiplayer online games: Minigame CAPTCHAs. In *Cyberworlds (CW), 2010 International Conference on*, 254 –261.

Gianvecchio, S.; Wu, Z.; Xie, M.; and Wang, H. 2009. Battle of Botcraft: fighting bots in online games with human observational proofs. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, 256–268. New York, NY, USA: ACM.

Golle, P., and Ducheneaut, N. 2005. Preventing bots from playing online games. *Comput. Entertain.* 3:3–3.

Grossman, T., and Balakrishnan, R. 2005. A probabilistic approach to modeling two-dimensional pointing. *ACM Trans. Comput.-Hum. Interact.* 12:435–459.

Ntzoufras, I. 2009. *Bayesian Modeling Using WinBUGS*. John Wiley & Sons.

Yampolskiy, R. V., and Govindaraju, V. 2008. Embedded noninteractive continuous bot detection. *Comput. Entertain.* 5:7:1–7:11.

Yan, J., and Randell, B. 2005. A systematic classification of cheating in online games. In *Proceedings of 4th ACM SIGCOMM workshop on network and system support for games*, NetGames '05, 1–9. New York, NY, USA: ACM.