Watermarking Game-Playing Agents in Perfect-Information Extensive-Form Games

Juho Kim
Computer Science Department
Carnegie Mellon University
juhok@cs.cmu.edu

Fei Fang
Software and Societal Systems Department
Carnegie Mellon University
feifang@cmu.edu

Tuomas Sandholm
Computer Science Department, CMU
Strategic Machine, Inc.
Strategy Robot, Inc.
Optimized Markets, Inc.
sandholm@cs.cmu.edu

Abstract

Watermarking techniques for large language models (LLMs), which encode hidden information in the output so its source can be verified, have gained significant attention in recent days, thanks to their potential capability to detect accidental or deliberate misuse. Similar challenges involving model misuse also exist in the context of game-playing, such as when detecting the unauthorized use of AI tools in gaming platforms (*e.g.*, cheating in online chess). In this paper, we initiate the study of how game-playing strategies can be watermarked. We show how the KGW watermark for LLMs can be adapted to watermark game-playing agents in perfect-information extensive-form games. The watermark can then be detected using a statistical test. We show that the degradation in the quality of the watermarked strategy profile, quantified by the expected utility, can be bounded, but there is a tradeoff between detectability and quality. In our experiments, we bootstrap the watermarking framework to engines and solvers for games like chess, Go, gomoku, Connect Four, and tic-tac-toe and demonstrate that a) the impact of the watermark on the quality of the strategy is negligible and b) the watermark can be detected with just a handful of games.

1 Introduction

In the past decades, breakthroughs in the field of artificial intelligence (AI) led to the development of superhuman AI agents capable of beating top human experts in games like chess [3], Go [18], poker [1, 2], and dark chess [23]. As strong as they are in performance, game-playing AI agents can potentially grow even stronger as new algorithms develop and computing power increases. As such, there have been growing concerns about their unauthorized usage, presenting the need for the means to embed and detect a robust, unique signature, *i.e.*, a watermark, in their gameplay. Despite the lack of attention from the AI research community, detecting the misuse of AI agents in game-theoretic settings is important because many real-life interactions between rational, self-interested parties in politics, economics, operations research, *etc.* can be modeled by game theory.

Watermarking techniques can be useful to communities for games like chess and poker, which face a rampant **cheating** problem [6, 8], especially online, as it can be used to detect improper use of AI. Additionally, the developers of AI tools themselves can use a watermark for license enforcement

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and **intellectual property (IP) protection**. In 2021, the development team behind the open-source chess engine Stockfish filed a lawsuit against the distributor of Houdini 6, a proprietary chess engine, accusing them of code plagiarism and license non-compliance [21]. One central piece of evidence included Houdini's leaked source code; however, had a hidden watermark been used beforehand, they would have had another, potentially more irrefutable piece of evidence of IP theft. In another use case, AI researchers interested in mimicking human behavior in games can use a watermark detector for **data cleaning** to sanitize datasets that are contaminated with AI gameplay.

There exists a vast literature [7, 12] on watermarking texts generated by large language models (LLMs), that is, encoding hidden signals in LLM outputs so their source can be verified. In this paper, we propose adapting LLM watermarks for game playing. Specifically, we explore adapting the KGW watermark, which is the most prominent and widely established method (later in our discussions, we justify this choice and discuss adapting alternative LLM watermarks). LLM watermarks were developed in response to the meteoric rise of LLMs in popularity and their strong performance in broad natural language processing tasks to address concerns about the potential risks of their accidental or deliberate misuse. Real-life examples of this include LLM-operated bot networks on social media platforms [14], large-scale phishing scams [16], and the spread of misinformation [20]. Additionally, revelations about the collapse of AI models when trained repeatedly on synthetic data [17] present the necessity to detect LLM-generated outputs in LLM training sets.

Curiously, text generation and game-playing have **striking similarities**, as summarized in Table 1, which allow LLM watermarks to be readily adapted to game-playing. Just as LLMs model the probability distribution over tokens given a context or a prompt as a prior, a strategy profile provides the probability distribution over the set of available actions at a particular history or information set (infoset). The text generation process ends upon sampling the end-of-sequence (EOS) token. This is analogous to how game-playing agents apply actions until they reach a terminal history.

In LLM watermarking, a watermark distorts the LLM's modeled probability distribution and thus can potentially degrade the quality of the generated text, typically quantified by calculating the perplexity cost [12] or conducting user feedback studies [7]. Naturally, such a concern carries over to game playing as well, where a watermark's distortion of action probabilities can reduce an agent's expected utility or make the agent vulnerable to exploitation.

Table 1: Analogies between text generation and game playing.

Text generation		Game playing
LLM	$\longleftrightarrow$	Strategy profile
Token	$\longleftrightarrow$	Action
Context/Prompt	$\longleftrightarrow$	History/Infoset
EOS token	$\longleftrightarrow$	Terminal history
Text quality	$\longleftrightarrow$	Expected utility /Exploitability
Sentence entropy	$\longleftrightarrow$	Mobility/Purity

There are certain situations in text generation where watermarking must be forgone, namely when dealing with low sentence entropy. As an example, consider the Python code snippet '`print("Hello,`', which is most likely followed by '`World!")`'. Here, the LLM watermark should more or less preserve the LLM's original probability distributions. Similarly, a watermarked strategy profile should retain its original strategy in situations with low mobility or when the action probability distribution is pure or almost pure. A real-life example of this would be in chess openings or endgames, where the set of correct actions is limited and playing incorrect actions can be detrimental.

However, **notable differences** exist between text generation and game playing. Whereas in text generation, a single LLM generates the text, in game-playing, multiple players, each employing their own strategies, apply actions sequentially. Also, while an attacker in text generation can attempt to remove the watermark as a post-processing step, an analogous attack in game playing is impractical because a) undoing an action is usually not allowed in competitive settings, b) other players and, if present, a neutral observer can record the gameplay, and c) utilities are typically realized to players as soon as the game ends. In short, offline watermark removal attacks, although technically possible, are not relevant in game playing. Therefore, the attacker is limited to online attacks only, that is, removing the watermark for a single pending action each time the attacker is in turn to act.

That said, game playing also introduces **new challenges** that are not present in text generation. While a typical LLM user may be satisfied with simply having the tokens sampled to them, a user of a game solver or an engine may not be content with simply having actions sampled to them. Indeed, they may also be interested in exploring alternative actions and comparing them with the recommended

action using (estimated) expected utilities. Thus, for our use case, the watermark must be expanded so it can also return the associated expected utilities for different actions or histories.

1.1 Our contributions

In this paper, we initiate the study of watermarking strategies for gameplay. We show how the widely established KGW watermark technique [12] for LLMs can be adapted for perfect-information extensive-form games. We also describe how to address the new challenges introduced by this setting, namely, returning the (estimated) expected utilities without compromising the watermark. We show that, although watermarking can distort the action probability distribution, the degradation in the quality of the watermarked strategy profile, quantified by the expected utility, can be bounded; however, there exists a tradeoff between detectability and quality. In our experiments, we bootstrap the watermark to mainstream engines and solvers for games like chess, Go, gomoku, Connect Four, and tic-tac-toe to a) demonstrate that the Elo differences between the game engines and their watermarked counterparts are negligible, b) show that the watermark can be detected with just a handful of games, and c) ablate on the watermark parameters.

2 Background and related work

We review game-theoretic concepts, LLM watermarking, and cryptographic applications to games.

2.1 Perfect-information extensive-form games (EFGs)

A *perfect-information extensive-form game (EFG)* contains a set of players P, which includes the chance player p_c, and a set of histories H. A history is a sequence of actions, and each history h is associated with a player in turn to act $p(h)$ and a set of available actions $A(h)$. We use $h \cdot a$ to denote applying $a \in A(h)$ at h. Also, we use $h \sqsubseteq h'$ to denote that h is a prefix of h'. When $p(h) = p_c$, *i.e.*, h is a chance history, then $f_c(h, \cdot)$ gives a probability distribution over $A(h)$. $Z \subseteq H$ is a set of terminal histories. Every terminal history $z \in Z$ is associated with $|P|$ utilities (one for each player), and we use $u_i(z)$ to denote the value for $i \in P$.

Each player $i \in P$ plays with strategy σ_i that assigns a probability distribution over the set of available actions in every history h where $p(h) = i$. A strategy profile σ is defined as a $|P|$-tuple of strategies from all players. We denote by $u_i(\sigma)$ the expected utility of i when players play according to σ. We use σ_{-i} to denote the strategies of all players except that of i. A best response of i to σ_{-i} is denoted as $BR(\sigma_{-i})$ and satisfies $u_i(BR(\sigma_{-i}), \sigma_{-i}) = \max_{\sigma'_i} u_i(\sigma'_i, \sigma_{-i})$. A traditional solution concept in perfect-information EFGs is a subgame perfect Nash equilibrium (SPNE) where no player stands to gain by best responding individually, even in states reached with zero probability.

An SPNE can be calculated using backward induction or approximated using algorithms such as alpha-beta pruning. Backward induction is a dynamic programming technique that propagates game utilities from the leaves to the root of the game tree. At each parent node, the corresponding player selects a child node that yields the maximum expected utility and sets the parent's values to the child's. In the alpha-beta pruning family of algorithms, the expected utility is replaced with its heuristic estimate after a certain depth, and pruning is performed on the game tree to cut down the number of explored nodes. Typically, the heuristic function is either hand-crafted or is implemented using a neural network trained via reinforcement learning.

2.2 Watermarking LLMs

We begin by formalizing *text generation*. Let $\mathcal{V}$ be the vocabulary, *i.e.*, the set of tokens (words or parts of words). An LLM is designed so that, given a sequence of tokens $v_1, v_2, \ldots, v_{t-1} \in \mathcal{V}$ as a prompt, it outputs a probability distribution $\vec{p} \in \Delta^{\mathcal{V}}$ over $\mathcal{V}$. From $\vec{p}$, the next token v_t is sampled, which is then appended to the input sequence. The resulting sequence is passed back to the LLM to obtain v_{t+1}. The text generation process continues in a loop until an EOS token is encountered.

While there are several methods to watermark LLM outputs [7, 12, 5], we center our focus on the widely established *KGW watermark*. The KGW watermark is flexible in that it can not only be bootstrapped to an existing LLM as a proxy but also be made public, *i.e.*, all of its parameters can be released to the public, allowing any interested third parties to detect the watermark without any access

Algorithm 1: KGW watermark adapted for game playing

Input: strategy profile σ, **green** list size $\gamma \in (0, 1)$, and hardness $\delta \geq 0$.

NEXTACTION(history $h \in H$):

  Query the underlying model of σ for the expected utilities $\vec{u} \in \mathbb{R}^{A(h)}$ over available actions at h.

  Seed a pseudo-random number generator with the hash of the observation made at h.

  Randomly partition $A(h)$ into a **green** list G of size $\gamma|A(h)|$ and a **red** list R of size $(1-\gamma)|A(h)|$.

  $\vec{v} \coloneqq \vec{u} + \left(\begin{cases} \delta & a \in G \\ -\gamma\delta/(1-\gamma) & a \in R \end{cases} \right)_{a \in A(h)}$;

  $a^* \coloneqq \arg\max_{a \in A(h)} \vec{v}_a$;

  return a^*;

to the LLM. Optionally, the watermark can be employed privately with a secret key, so the detection service can be provided through a secure API. In addition, the KGW watermark can be detected with a statistical test. By adapting this watermark for game playing, we can inherit its desirable properties.

The pseudocode of the KGW watermark is available in Appendix A. It accepts the following parameters: a **green** list size $\gamma \in (0, 1)$ and hardness $\delta \geq 0$. The watermark begins by querying the LLM with the input prompt to obtain logits $\vec{l} \in \mathbb{R}^{\mathcal{V}}$ over the vocabulary. Then, the hash of v_{t-1} is used to seed a pseudo-random number generator. The generator is then used to randomly partition $\mathcal{V}$ into a **green** list G of size $\gamma|\mathcal{V}|$ and a **red** list R of size $(1-\gamma)|\mathcal{V}|$. The watermark then modifies $\vec{l}$ to introduce a bias toward selecting a token in G as opposed to R during the sampling process. This is achieved by adding δ to the logits corresponding to the tokens in G. Finally, $\vec{l}$ is passed to a softmax function to obtain probabilities $\vec{p} \in \Delta^{\mathcal{V}}$, from which v_t is sampled.

2.3 Steganography in game-playing

Watermarking is a special case of *steganography*, which encompasses methods of hiding secret data in a public medium such as text and images. One medium of secret communication explored by several prior works is gameplay data [4, 9, 13], but their techniques are tailored for encoding **generic** secret data, and thus are not specialized enough for watermarking. Plus, a notable difference between our contribution and the prior works of Mandujano et al. [13] and Chang and Echizen [4] is that their algorithms are not designed to be bootstrapped to existing player policies. Moreover, Mandujano et al. [13] control both the player and the game environment (*i.e.*, the nature player), which is not representative of typical game-theoretic settings. Additionally, unlike ours, these prior works do not provide any game-theoretic guarantees, such as bounds on the loss in expected utility.

3 Methodology

We are now ready to present our results. We begin by discussing how one can adapt the KGW watermark [12] for strategy profiles in perfect-information EFGs.

3.1 Watermarking strategy profiles

Whereas, in text generation, the watermark acts as a proxy to an LLM; in game playing, it acts as a wrapper to a strategy profile σ, calculated by an underlying game-solving algorithm. Akin to querying the LLM for logits over the vocabulary, our watermark queries the underlying model of σ with history h to calculate (or estimate) the expected utilities over the set of available actions $A(h)$.

We then use the hash of the observation made at h to seed the pseudo-random number generator. This deviates from the original KGW algorithm, which used the final token of the prompt (an analogue of the final action) to seed the pseudo-random number generator. We begin by justifying why we **can** make such a change. In the original KGW watermark, the motivation for seeding with the final token

Algorithm 2: Watermark for expected utility

Input: strategy profile σ, **green** list size $\gamma \in (0, 1)$, and hardness $\delta \geq 0$.

EXPECTEDUTILITY(history $h \in H$):
  Query the underlying model of σ for the expected utility u at h.
  $v \coloneqq u$;
  for *each* $h' \cdot a \sqsubseteq h$ **do**
    if NEXTACTION*(*h'*)* *partitions* $A(h')$ *s.t.* $a \in G$ **then**
      $v \coloneqq v + \delta$;
    else
      $v \coloneqq v - \gamma\delta/(1-\gamma)$;
  return v;

was to make the watermark robust against splicing attacks and hence remain detectable even with a contiguous substring of the generated text. In game-playing, this consideration is no longer relevant, as splicing attacks are impossible due to the reasons stated in the introduction (*e.g.*, presence of a neutral/adversarial observer, impossibility of offline attacks, *etc.*).

We continue by explaining why we **need** to make such a change. Applying the approach of Kirchenbauer et al. [12] directly in game playing can be dangerous, as the attacker, trying to get a recommendation for h, can pass another history h' with a different final action that nonetheless produces the same observation. By doing so, the attacker would be able to get a recommendation for h while bypassing the watermark. That said, using the observation at h instead of the history itself is not fully robust either, as the attacker can instead pass another history h'' that results in a similar albeit different observation. While the attacker can theoretically bypass the watermark this way, later in the paper, we discuss why this is risky for the attacker and propose how such an attack can be countered.

Then, we use the pseudo-random number generator to randomly partition the available actions into a **green** list G of size $\gamma|A(h)|$ and a **red** list R of size $(1-\gamma)|A(h)|$, and add δ to the (estimated) expected utilities corresponding to the actions in G while subtracting $\gamma\delta/(1-\gamma)$ from those in R. Here, unlike the KGW watermark, we also subtract from the values corresponding to the **red**-list actions to ensure that the adjustment to every (estimated) expected utility is zero in expectation. Finally, the action yielding the maximum adjusted (estimated) expected utility is selected, which is then returned. The pseudocode of the watermarking process is shown in Algorithm 1. When $\delta \to \infty$, the watermark only returns **green**-list actions, and when $\delta \to 0^+$, the watermark only plays a tiebreaking role.

3.2 Detecting the watermark

As in the KGW watermark, the null hypothesis is that a particular player applies actions with no knowledge of the **green** and **red** lists. Let n be the total number of actions from the player and n_G be the number of **green**-list actions from the player. We have the following z-score for an arbitrary γ:

$$z = \frac{n_G - \gamma n}{\sqrt{n\gamma(1-\gamma)}}.$$

The z-score reflects how many more (or fewer) **green**-list actions are applied than the number expected from a player oblivious to the watermark. Kirchenbauer et al. [12] suggested the z-score threshold of $z = 4$, corresponding to approximately 3×10^{-5} probability of getting a false positive.

3.3 Watermarking expected utilities

The main watermark introduced above can be thought of as a black box that takes a history as an input and outputs a **recommended action**. As alluded to in the introduction, in game playing, a user may not be satisfied with just being provided with the recommended action. Indeed, they may also want the associated expected utility for the recommendation. Moreover, they may want to compare the expected utility of applying this action with that of applying an alternative action as well. This presents a need for another watermark that takes a history as an input and outputs its **expected value**.

One cannot simply return the (estimated) expected utility from the underlying model of σ without any modification, as an attacker would be able to bypass the watermark by picking an action with the maximum expectation. A sensible option is to modify the expected utility with the hardness parameter δ as shown in Algorithm 2. This is consistent with our main watermark (shown in Algorithm 1).

However, when this watermarking algorithm is public and δ is known to the attacker, the attacker would be able to undo the watermark adjustments to obtain the true expected utility returned by the underlying model. As a consequence, if information about the expected utility is provided to users, the watermark must be deployed privately. This caveat, although not explored by Kirchenbauer et al. [12], is also highly relevant in text generation: the KGW watermark must be deployed privately when, alongside sampled tokens, the logits or the probability distributions over the vocabulary are also returned. The expanded version of the conference paper by Kirchenbauer et al. [12] discusses how to deploy the KGW watermark privately, and the same technique can be applied to game playing.

4 Analysis

In this section, we analyze the KGW watermark adapted to game playing. For the purpose of analysis, we assume that a) the underlying model of the strategy profile being watermarked gives the true expected utility, b) whether any chosen action is in the **green** or **red** list is independent of the types of other chosen actions, and c) the probability of a watermarked agent selecting a **green**-list action is constant. Then, we have the following theorem:

Theorem 1. *Given a strategy profile σ, let σ^* be its watermarked counterpart, following Algorithm 1, and suppose that an agent follows σ^*. Furthermore, suppose that whether the watermarked agent chooses a **green**- or **red**-list action at any decision point is independent of the choices made at other decision points, and that the probability of the agent selecting a **green**-list action is a constant p. Let n be the number of actions played by the agent, and let L be the total loss in utility the agent incurs by applying the watermark against a fixed opponent. For any $t > 0$,*

$$\Pr[L < \delta(1 + \gamma/(1-\gamma))(pn + t)] > 1 - e^{-2t^2/n}.$$

The proof is in Appendix B.1. Relaxing the independence assumption and the assumption that the probability of selecting a **green**-list action is constant, we have the following proposition for the deterministic bound on the loss in utility an agent incurs by applying the watermark:

Proposition 1. *Given a strategy profile σ, let σ^* be its watermarked counterpart, following Algorithm 1, and suppose that an agent follows σ^*. Let n be the number of actions played by the agent, and let L be the total loss in utility the agent incurs by applying the watermark against a fixed opponent. Then,*

$$L \leq n\delta(1 + \gamma/(1-\gamma)).$$

The proof is in Appendix B.2. In general, there exists a **tradeoff** between detectability and quality in the watermarked strategy profile. While increasing δ can help detect the watermark more quickly, *i.e.*, with fewer actions played, doing so causes the agent to potentially lose out on the expected utility.

While the potential loss in expected utility incurred by applying the watermark may be concerning, it may not be as relevant when the underlying model of σ only **estimates** the expected utility (which applies to solvers for large perfect-information EFGs), since the loss in the true expected utility from the inaccuracy of the model can be much greater than the loss incurred by applying the watermark. In this regard, the watermark can be viewed as simply adding tiny additional noise to the already noisy estimates. Later in our experiments, we empirically demonstrate that the performance degradation in major chess engines due to watermarking is insignificant.

However, if the game tree depth is extremely long (*e.g.*, thousands of actions), it is possible that watermarking can severely hamper the watermarked agent's gameplay. In watermarking agents playing such games, we suggest modifying the watermark so that watermarked actions are only applied to a fixed number of actions. For instance, in what we call the early T-stopping watermark, the watermark can only be applied to the first T actions for some fixed T. Theoretically, the maximum loss in expected utility incurred by the early-stopping watermark would be a manageable constant. Later, we verified this idea in our ablations.

5 Experiments

One notable property of the KGW watermark is that it can easily be bootstrapped to existing LLMs. The same applies to the watermark when adapted from text generation to game playing. In our experiments, we bootstrapped the watermark to six selections of popular universal chess interface (UCI) engines: asmFish 9, Dragon, Ethereal 14.40, PlentyChess, Stash v37.0, and Stockfish 17.1. To calculate the degradation in the quality of the strategy profile introduced by watermarking, as quantified by the loss in expected utility, we pitted each chess engine against its watermarked counterpart. We seek to demonstrate that a) the watermark does not significantly harm the chess engine performance and b) the watermark can be detected after relatively few rounds of play.

In our experiments, each match consisted of 100 rounds (*i.e.*, complete games), beginning from 8-move positions in the Stockfish chess books dataset, under the time control of 40 moves in 60 seconds. We calculated the Elo difference resulting from the watermark, as well as the likelihood that the watermarked engines are inferior in strength to their original counterparts. We also calculated draw probabilities and z-scores from the gameplay data. Making UCI engines output multiple actions typically incurs some computational cost. To keep a level playing field, we used the same engine parameters for both watermarked and non-watermarked engines. The watermark parameters were set as $\gamma = 0.25$ and $\delta = 0.5$ centipawn, which we later found to be ideal in our ablation studies.

The experimental results are tabulated in Table 2. Even after 100 rounds, the Elo differences between the watermarked and non-watermarked chess engines are well within the margins of error. Owing to the stochastic nature of computer chess matches, several watermarked engines are seen to even outperform their non-watermarked counterparts. Additionally, for every chess engine, the likelihood that its watermarked counterpart is inferior (LOI) fails to reach the 90% confidence level. We interpret these results as empirical evidence that the watermark does not significantly harm performance.

The z-scores of the watermarked engines, calculated from the entire match, are far above the suggested $z = 4$ threshold, whereas those of the non-watermarked engines do not meet the threshold. Recall that $z = 4$ corresponds to the false-positive rate of approximately 3×10^{-5}. Figure 1 shows how the z-scores change over the number of rounds played. It can be seen that the z-scores of the watermarked chess engines cross the suggested $z = 4$ threshold relatively early into the match. The Dragon chess engine passes the threshold only after playing a single round, while asmFish 9 and Stockfish 17.1 do so after finishing Round 2. The rest hit the threshold after Rounds 4, 8, and 9.

Having watermarked chess engines reach the z-threshold in just a few rounds can be advantageous to a game-playing platform for the sake of security and ecology, as it limits the potential harm inflicted on other legitimate players. That said, the data suggest that monitoring a potentially malicious actor for a single round may not be enough; instead, multiple rounds must be played to ascertain whether a player is cheating with statistical significance. Indeed, for each round played, we swept the z-score thresholds and found that the area under the receiver operating characteristic (ROC) curves (AUCs) averaged between 0.7 and 0.8, which is not ideal. The ROC curves and their AUC values are in Figure 1 and Table 2. In our ablation studies, we show that the AUC values can be increased by sacrificing the engine's strength. Additional experiments on other games like Go, gomoku, Connect Four, and tic-tac-toe are in Appendix D.

6 Ablation Studies

In our ablation studies, we exclusively use the Stockfish 17.1 chess engine. First, fixing the hardness to $\delta = 0.5$, we iterated over different values of the **green**-list size γ. For each set of parameters, we initialized the watermarked chess engine and recorded its match of 100 rounds against the original Stockfish engine. In our second ablation study, we fixed $\gamma = 0.25$ and iterated over different values of δ. Again, we recorded the statistics of the matches played.

The results are tabulated in Table 3. When iterating over the **green**-list size γ (with fixed hardness $\delta = 0.5$), it can be seen that $\gamma = 0.25$ results in the maximum z-score of the watermarked chess engine. It also results in the maximum AUC of the ROC curve. Our results show that setting $\gamma = 0.25$ is optimal. More monotonicity is observed when iterating over the hardness δ, with fixed **green**-list size $\gamma = 0.25$. As δ increases, the LOI approaches 1, meaning that the watermarked engine is significantly weaker in strength compared to the original Stockfish engine. An increase in δ also corresponds to an increase in the z-score of the watermarked engine and the AUC of the ROC curve.

Table 2: Results from the UCI engine matches ($\gamma = 0.25$, $\delta = 0.5$ centipawn). A negative Elo difference denotes that the watermarked engine is weaker and vice versa. LOI is the likelihood that the watermarked engine is weaker than the original. $\#_{z=4}$ denotes the number of games taken for the $z = 4$ threshold to be reached.

Engine	Elo difference	LOI	# draws	z-score NW	z-score W	$\#_{z=4}$	AUC
asmFish 9	-10.4 ± 55.3	64.5%	35	-2.14	$+22.9$	2	0.793
Dragon	$+10.4\pm47.9$	33.4%	51	$+0.151$	$+47.0$	1	0.790
Ethereal 14.40	-10.4 ± 52.6	65.2%	41	-0.865	$+33.4$	8	0.783
PlentyChess 7.0.0	-6.9 ± 33.4	65.8%	76	$+0.262$	$+19.4$	4	0.715
Stash v37.0	-20.9 ± 44.3	82.3%	58	-0.0464	$+28.8$	9	0.787
Stockfish 17.1	$+24.4\pm31.0$	6.3%	79	$+0.894$	$+28.8$	2	0.766

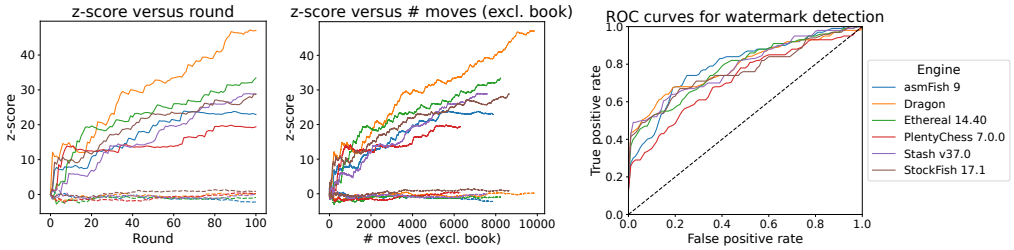


Figure 1: On the left and in the middle, the z-scores over the number of rounds and moves, respectively. On the right, the ROC curves for watermark detection (for each round). Solid and dotted lines represent watermarked and original chess engines, respectively. The AUC values of the ROC curves are tabulated in Table 2.

Table 3: Results from the Stockfish chess engine matches under parameter ablation. The value should be interpreted as in Table 2. The asterisked values (with *) are from the same match and are also tabulated in Table 2 as entries for Stockfish.

γ	δ	Elo difference	LOI	# draws	z-score NW	z-score W	$\#_{z=4}$	AUC
0.1	0.5	$+17.4\pm36.7$	17.7%	71	-0.665	$+7.87$	37	0.671
0.25	0.5	$+24.4\pm31.0^*$	$6.3\%^*$	79^*	$+0.894^*$	$+28.8^*$	2^*	0.766^*
0.5	0.5	-20.9 ± 30.3	91.0%	80	-0.693	$+14.7$	6	0.728
0.75	0.5	-3.5 ± 34.1	57.9%	75	$+0.539$	$+6.60$	25	0.656
0.9	0.5	$+6.9\pm37.4$	35.8%	70	-1.41	$+2.10$	>100	0.592
0.25	0.5	$+24.4\pm31.0^*$	$6.3\%^*$	79^*	$+0.894^*$	$+28.8^*$	2^*	0.766^*
0.25	1	$+10.4\pm32.7$	26.6%	77	$+0.950$	$+26.2$	5	0.760
0.25	2	$+17.4\pm29.6$	12.6%	81	$+0.224$	$+34.3$	8	0.850
0.25	5	-24.4 ± 39.1	88.8%	67	-0.376	$+42.4$	2	0.917
0.25	10	-88.7 ± 41.7	100.0%	61	$+0.600$	$+40.5$	2	0.962

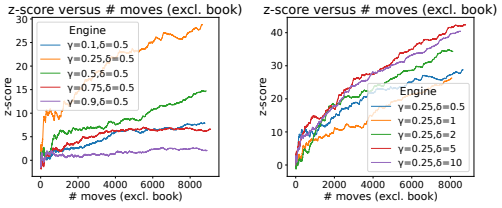


Figure 2: z-scores for watermarked Stockfish 17.1. The left and right plots ablate on γ and δ, respectively.

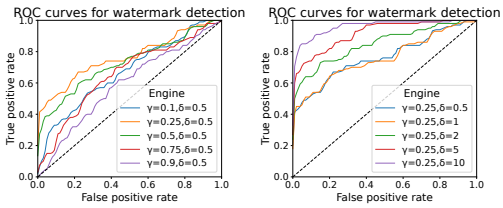


Figure 3: The ROC curves for watermark detection (for each round). The AUC values are tabulated in Table 3.

Clearly, increasing δ facilitates the watermark detection, but it also diminishes the engine's strength. Additional ablations to study our T-stopping variant are in Appendix E.

7 Discussion

Here, we justify the adaptation of the KGW watermark and discuss possible attacks and extensions.

7.1 Use of alternative LLM watermarks

The primary reason why we chose the KGW watermark for our study was that its logit boosting mechanism makes the watermark particularly applicable to game-playing settings, as boosting the analogue of logits for playing perfect-information EFGs – expected utilities – is both simple and intuitive, and, as we have shown in this paper, leads to a highly performant watermark for game-playing agents. Similarly, we would expect the variants [24, 19, 11, 22] of the KGW watermark that rely on the core idea of partitioning tokens into **red**- and **green**-lists to remain highly applicable to the game-playing setting, as long as the newly introduced ideas have obvious analogues in the game-playing setting. For example, LTW introduced by Wang et al. [22] uses sentence embeddings, whose analogues in game playing are history embeddings. That said, there are also LLM watermarks such as SynthID [7], which operate on tokens sampled from existing LLMs. Since solutions to perfect-information EFGs give pure probability distributions over available actions, the watermark would sample the same action repeatedly; therefore, SynthID and other similar watermarks will fail when adapted to game playing, although a smoothing method can possibly be used to overcome this.

7.2 Attacking the watermark

One advantage of adapting the KGW watermark for use in game playing is that some of its robustness guarantees can be inherited. This is compounded by the fact that, as we argued in the introduction, game playing prohibits the attacker from modifying previously played actions. Thus, common offline watermark attack patterns such as the family of paraphrasing, insertion, and replacement attacks are not relevant in this setting. With that said, we tailored the KGW watermark for the purpose of playing perfect-information games by making a number of modifications to the original algorithm. In this section, we argue that our changes do not facilitate an attacker in carrying out online attacks, which involve the attacker manipulating the watermark each time the attacker is in turn to act.

A notable difference between the KGW watermark and the modified watermark is that we use the observation made at the queried history instead of the last action (or token) to seed the pseudo-random number generator. To manipulate this, the attacker, prior to querying the watermarked strategy profile, could try to modify the history so that the resulting observation is slightly different from the original observation. This is risky for the attacker, as, in games like chess, even small differences in piece positions that seem benign at first glance can have large strategic implications at play. As a result, this attack can severely degrade the quality of the recommendations, which goes against the interest of the attacker. Also, to counter such an attack, locality-sensitive hashing can be used so that similar observations result in identical hash values. Since the hash values are identical, the set of available actions will be partitioned identically across similar observations, thus nullifying the attack.

Henceforth, we make the standard assumption [12] that the watermarked strategy profile is deployed privately. Instead of taking the recommendation from the watermarked strategy profile, the attacker could try to obtain the watermarked (estimated) expected utility for each available action and use it during action selection. One possible strategy is selecting the top-k available actions or those above a certain threshold, from which an action is further subsampled to be played. This strategy will fail, as the attacker's process of sampling the top actions is still biased toward sampling actions in the **green** list. Among the sampled actions, the attacker has no way of discriminating between **red**- and **green**-list actions, so the watermark will remain detectable in the attacker's gameplay no matter what subsampling strategy is used. In fact, any strategy that is biased toward selecting actions with higher watermarked (estimated) expected utilities would remain detectable.

To circumvent the watermark, the attacker could avoid using the watermarked strategy profile except in a few crucial situations. However, as long as the attacker uses the watermarked strategy profile, the gameplay will show that **green**-list actions are played with probability higher than γ. As a result,

the z-score calculated from the attacker's gameplay will eventually cross the designated z-threshold. Thus, the watermark will remain detectable, although its detection will require more gameplay data.

8 Conclusions and future research

In this paper, we initiated the study of how strategies can be watermarked. We showed how the widely established KGW watermark [12] for LLMs can be adapted to watermark strategy profiles in perfect-information games. Game playing introduces new challenges not present in text generation, namely having the (estimated) expected utilities provided to the user, along with the recommendation. We showed how the information about (estimated) expected utilities can be watermarked as well. Also, in our analysis, we bounded the loss in the expected utility introduced by the watermark. In our experiments, we bootstrapped the watermark onto prominent engines and solvers for games like chess, Go, gomoku, Connect Four, and tic-tac-toe and showed that no significant degradation in the engine's strength can be observed. Then, we discussed possible attacks on the watermark and proposed countermeasures. Our method can also be applied to single-agent decision making, as it is a special case of perfect-information EFGs. Possible future work includes adapting our watermark for antidistillation fingerprinting [15] and extending it to imperfect-information EFGs. Imperfect-information settings require one to mix their strategies in order to be optimal, which introduces new challenges not addressed in this paper. One would need to bound the increase in exploitability due to the watermark.

Acknowledgements

This work has been supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876, National Science Foundation grant RI-2312342, and NIH award A240108S001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

The authors have no relevant competing interests to declare.

The authors would like to thank the anonymous reviewer for suggesting the early-stopping idea.

References

[1] N. Brown and T. Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[2] N. Brown and T. Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[3] M. Campbell, A. J. Hoane, Jr., and F. Hsu. Deep Blue. *Artificial Intelligence*, 134(1):57–83, 2002.

[4] C.-C. Chang and I. Echizen. Steganography in game actions. *IEEE Access*, 13:21029–21042, 2025.

[5] Y. Chang, K. Krishna, A. Houmansadr, J. F. Wieting, and M. Iyyer. PostMark: A robust blackbox watermark for large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[6] B. Chappell. Hans Niemann is accused of cheating in more than 100 chess games. He's playing today. *National Public Radio (NPR)*, 2022.

[7] S. Dathathri, A. See, S. Ghaisas, P.-S. Huang, R. McAdam, J. Welbl, V. Bachani, A. Kaskasoli, R. Stanforth, T. Matejovicova, J. Hayes, N. Vyas, M. A. Merey, J. Brown-Cohen, R. Bunel, B. Balle, T. Cemgil, Z. Ahmed, K. Stacpoole, I. Shumailov, C. Baetu, S. Gowal, D. Hassabis, and P. Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634 (8035):818–823, 2024.

[8] C. Grant. Poker pro axed from sponsorship after using solver while playing online. *PokerNews*, 2026.

[9] J. C. Hernandez-Castro, I. Blasco-Lopez, J. M. Estevez-Tapiador, and A. Ribagorda-Garnacho. Steganography in games: A general methodology and its application to the game of go. *Computers & Security*, 25(1):64–71, 2006.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[11] M. Huo, S. A. Somayajula, Y. Liang, R. Zhang, F. Koushanfar, and P. Xie. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[12] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

[13] R. Mandujano, J. Gutierrez-Cardenas, and M. S. Monge. Steganography application using combination of movements in a 2d video game platform. In K. Arai, S. Kapoor, and R. Bhatia, editors, *Proceedings of the Future Technologies Conference (FTC)*, 2020.

[14] B. Qiao, K. Li, W. Zhou, S. Li, Q. Lu, and S. Hu. BotSim: LLM-powered malicious social botnet simulation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

[15] T. Sander, P. Fernandez, A. O. Durmus, M. Douze, and T. Furon. Watermarking makes language models radioactive. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[16] J. Schultz. Cybercriminal abuse of large language models. *Cisco Talos*, 2025.

[17] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, and Y. Gal. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

[19] Y. Takezawa, R. Sato, H. Bao, K. Niwa, and M. Yamada. Necessary and sufficient watermark for large language models. *Transactions on Machine Learning Research*, 2025.

[20] The Lancet Digital Health. Large language models and misinformation. *The Lancet Digital Health*, 8(1):1, 2026.

[21] The Stockfish Team. Our lawsuit against ChessBase, 2021.

[22] C. Wang, J. Shu, B. Chiu, Y. Li, S. Alharbi, M. Zhang, and J. Li. Learning to watermark: A selective watermarking framework for large language models via multi-objective optimization. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[23] B. H. Zhang and T. Sandholm. General search techniques without common knowledge for imperfect-information games, and application to superhuman Fog of War chess. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

[24] X. Zhao, P. V. Ananth, L. Li, and Y.-X. Wang. Provable robust watermarking for AI-generated text. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

A Pseudocode of the KGW watermark

The pseudocode of the KGW watermark is shown in Algorithm 3. A textual explanation of the pseudocode is available in Section 2.2.

Algorithm 3: KGW watermark for text generation

Input: LLM with vocabulary $\mathcal{V}$, **green** list size $\gamma \in (0,1)$, and hardness $\delta \geq 0$.

NEXTTOKEN(prompt $v_1, v_2, \ldots, v_{t-1} \in \mathcal{V}$):

  Query the LLM with prompt $v_1, v_2, \ldots, v_{t-1}$ to obtain logits $\vec{l} \in \mathbb{R}^{\mathcal{V}}$ over the vocabulary.

  Seed a pseudo-random number generator with the hash of v_{t-1}.

  Randomly partition $\mathcal{V}$ into a **green** list G of size $\gamma|\mathcal{V}|$ and a **red** list R of size $(1-\gamma)|\mathcal{V}|$.

  $\vec{p} := \text{softmax}\left(\vec{l} + \left(\begin{cases}\delta & v \in G \\ 0 & v \in R\end{cases}\right)_{v \in \mathcal{V}}\right)$;

  Sample the next token $v_t \in \mathcal{V}$ from $\vec{p}$.

  return v_t;

B Omitted proofs

This section contains the proofs omitted from the main body of the paper.

B.1 Proof of Theorem 1

Proof. We can formalize this setting as follows: let $X_1, \ldots, X_n \in \{0,1\}$ be independent random variables where, for each $i \in [n]$, $X_i = 1$ if the agent chooses a **green**-list action as the i^{th} action or $X_i = 0$ if otherwise. As per our assumption, we have that $\forall i \in [n] : \Pr[X_i = 1] = p$. Denote the number of times the agent selects a **green**-list action by $X = \sum_{i=1}^{n} X_i$. Note that

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \Pr[X_i = 1] = \sum_{i=1}^{n} p = pn.$$

By Hoeffding's inequality [10], we have that, for any $t > 0$,

$$\Pr[X \geq pn + t] < e^{-2t^2/n}.$$

The loss in utility the agent incurs by choosing a single **green**-list action is at most $\delta + \gamma\delta/(1-\gamma) = \delta(1 + \gamma/(1-\gamma))$. By definition, we have that $L \leq X\delta(1+\gamma/(1-\gamma))$, so $X < pn + t \implies L < \delta(1+\gamma/(1-\gamma))(pn+t)$. Then,

$$\Pr[L < \delta(1+\gamma/(1-\gamma))(pn+t)] \geq \Pr[X < pn + t] = 1 - \Pr[X \geq pn + t] > 1 - e^{-2t^2/n},$$

as required. □

B.2 Proof of Proposition 1

Proof. Let n_G be the number of **green**-list actions played by the agent. The loss in utility the agent incurs by choosing a single **green**-list action is at most $\delta + \gamma\delta/(1-\gamma) = \delta(1+\gamma/(1-\gamma))$. Since $n_G \leq n$, we have that

$$L \leq n_G\delta(1+\gamma/(1-\gamma)) \leq n\delta(1+\gamma/(1-\gamma)),$$

as required. □

Table 4: Results from the Go, gomoku, connect-four, and tic-tac-toe engine matches. The value should be interpreted as in Table 2. For connect-four and tic-tac-toe, the watermark only plays a tiebreaking role.

Game	Engine	Elo difference	LOI	# draws	z-score NW	z-score W	$\#_{z=4}$
Go	KataGo	$+20.9 \pm 69.1$	27.4%	0	-0.572	$+29.0$	2
	GNU Go	$+41.9 \pm 69.5$	11.5%	0	$+0.322$	$+8.10$	17
Gomoku	Rapfi	$+17.4 \pm 68.7$	30.8%	1	-0.527	$+14.9$	2
Connect four	–	–	–	0	-0.765	$+10.9$	3
Tic-tac-toe	–	–	–	100	-0.462	$+1.98$	>100

Table 5: Results from the Stockfish chess engine matches with the early T-stopping watermark. The values should be interpreted as in Table 2.

T	Elo difference	LOI	# draws	z-score NW	z-score W	$\#_{z=4}$
5	$+3.5 \pm 40.4$	43.3%	65	$+0.993$	$+2.01$	>100
10	$+3.5 \pm 29.7$	40.9%	81	$+1.22$	$+1.80$	>100
20	-10.4 ± 39.2	69.9%	67	$+0.170$	$+7.74$	32
50	-24.4 ± 36.6	90.3%	71	-0.178	$+19.6$	4
100	-10.4 ± 39.2	69.9%	67	-0.551	$+25.5$	4

C Testbench specification

For the experiments and ablation studies in the main body of the paper, our testbench contained an Intel Core i5-4690, a 4-thread desktop processor, and 16 GB of memory. For the additional experiments and ablation studies in the appendices, our testbench contained an AMD Ryzen 9 3900X 12-core, 24-thread desktop processor, 128 GB of memory, and an NVIDIA GeForce RTX 4090 graphics card.

D Additional Experiments

Continuing our experiments, we applied our watermark to other games such as Go, gomoku, connect-four, and tic-tac-toe. For Go, we watermarked the following engines: KataGo and GNU Go with $\delta = 0.05$ and used opening positions with 16 stones from the Fuego opening book. For gomoku, we watermarked the Rapfi engine with $\delta = 10$ and used the Gomocup 2025 opening book. For connect-four and tic-tac-toe, we created watermarked agents that use full game solutions with $\delta \to 0^+$ (*i.e.*, the watermark only plays a tiebreaking role) and used optimal depth-8 and -4 openings, respectively. For all games, we set $\gamma = 0.25$. For Go and gomoku engines, we allowed a thinking time of 1 second per stone. We matched each engine/solution against its watermarked counterpart for 100 games. The results are shown in Table 4. Again, the results show that the watermark is highly detectable (except for tic-tac-toe) and does not significantly harm the performance of these engines. For tic-tac-toe, we expect the watermarked engine's z-score to cross the $z = 4$ threshold when more games are played, as each game only allows 9 actions to be played in total.

E Additional Ablation Studies

Continuing our ablations, we tested the early T-stopping variant of our watermark by enumerating through $T \in \{5, 10, 20, 50, 100\}$ and playing 100 rounds of chess. Here, the watermark was used for the first T actions after the 8-move positions. The results are tabulated in Table 5. As expected, lowering the number of watermarked actions decreases the watermark's detectability, as more games are required for the $z = 4$ threshold to be crossed. In extreme cases where $T = 5$ or $T = 10$, even after 100 games, the $z = 4$ threshold was not crossed. However, for $T = 50$ and $T = 100$, it took only four games for the $z = 4$ threshold to be crossed. For every case but one, we found no statistical evidence that the original engine is stronger than the watermarked one with 90% confidence, which we can interpret as showing that watermarking does not significantly harm the agent's strength.

However, when $T = 50$, the probability that the watermarked engine was inferior is 90.3%. Given the broader trend across the main experiments and ablation studies, we attribute this isolated result to statistical variance.

F Additional Discussion

Here, we give some additional discussion on our watermark.

F.1 Early Stopping

While the early T-stopping strategy can bound the worst loss in expected utility by a manageable constant, more gameplay would be required for the watermark to be detected. In some settings, *e.g.*, online gaming platforms, this could be problematic in terms of platform security and ecology, as cheaters can ruin honest players' experience. Additionally, not all values of T may work in a game. Suppose that a game has a strategy that is clearly the best for the first 20 moves. If we set $T \leq 20$ and hardness to be very low, the watermarked agent and the original agent would behave identically, and thus it would be impossible to detect the watermark. Thus, setting the right value for T would require domain-specific knowledge or running experiments to identify which values of T work.

F.2 Handling large action spaces

It is possible that the strategy space may be too large to be partitioned, let alone enumerated. One effective way to address this concern would be to query the underlying model of the watermarked strategy profile for a handful of (say 25) recommended actions. Then, each action can be classified using the hash of the corresponding sequence (j, a), where j is the decision point, and a is the action. This approach is asymptotically equivalent to partitioning, so our theoretical results still hold.

It is also possible for games to have continuous action spaces, and this setting is quite different from text generation, which is discrete. For these types of games, one can convert them to standard EFGs by discretizing the action spaces. Alternatively, the continuous action space can be partitioned into shards of red and green regions. Then, the watermark can encourage the actions in the green regions to be more likely to be selected than those in the red regions.

F.3 Effects of game definitions

We now discuss how different games (*e.g.*, action space, game-tree topology, utility distribution, *etc.*) would affect our watermark. We expect the watermark to benefit from having larger action spaces, as a given situation is more likely to have some subpar actions that are almost as good as the best action. This is precisely the setting in which our watermarking technique excels, since green actions can be chosen more frequently with very little loss in expected utility. That said, our analysis predicts that the watermark may deteriorate in performance as the game-tree depth increases, but we can address this by applying early T-stopping. Additionally, if the expected utility estimators are poor, then the loss in expected utility incurred by the watermark would be negligible compared to that introduced by the estimators' inaccuracy. Thus, it would be **more difficult** to tell the difference in strength between the watermarked agent and its original counterpart. When the utilities are tightly distributed, we expect our watermark to do well since it would be able to select green actions more frequently, which allows us to decrease the hardness parameter, which, in turn, would decrease the theoretical loss in expected utility. In contrast, when the utilities are not tightly distributed, we expect our watermark to have a relatively harder time since green actions would be less frequently selected, and increasing this frequency would entail playing more subpar actions.

G Elo difference and LOI calculations

In this section, we present how the Elo difference and LOI values are calculated. Let W, L, and D be the number of wins, losses, and draws, and let $1 - \alpha$ be the desired confidence level, respectively. In this paper, we chose $\alpha = 0.05$. Let $N = W + L + D$ and $\mu = (W + D/2)/N$. Then, the estimated

Elo difference is defined as follows:

$$\Delta E = -400 \cdot \log_{10}\left(\frac{1}{\mu} - 1\right).$$

Now, we give the uncertainty calculations. Define the variance $\sigma^2 = (W(1-\mu)^2 + L(0-\mu)^2 + D(0.5-\mu)^2)/N$. Then, the standard error of the mean is $s = \sigma/\sqrt{N}$. Let $z = \Phi^{-1}(1-\alpha/2)$. Note that $z \approx 1.96$ for $\alpha = 0.05$. Then, the margin of error is as follows:

$$\text{PM} = \frac{\Delta E|_{\mu+zs} - \Delta E|_{\mu-zs}}{2}. \tag{1}$$

It is possible to obtain asymptotic bounds using the delta method, as follows:

$$\text{PM}_{\text{delta}} = \frac{400z\sigma}{\ln(10)\sqrt{N}\mu(1-\mu)} \tag{2}$$

The delta method (2) simplifies to $\text{PM}_{\text{delta}} = 400z/(\ln(10)\sqrt{N\mu(1-\mu)})$ when $D = 0$. As an example, this results in 68.2 and 68.6 for KataGo and GNU Go, respectively. For consistency, we only used (1) throughout the paper. The likelihood of superiority is as follows:

$$\text{LOS} = \frac{1 + \text{erf}((W-L)/\sqrt{2(W+L)})}{2}.$$

Then, the likelihood of inferiority is simply $\text{LOI} = 1 - \text{LOS}$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims made in the abstract and introduction are demonstrated by our experimental results and proven in Appendices B.1 and B.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We clearly state that the scope of our work is watermarking strategies in perfect-information EFGs and explicitly state the assumptions for Theorem 1 and Proposition 1. That said, we do not have a dedicated "Limitations" section in our paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The statements and their proofs (relegated to the appendix) are both numbered and cross-referenced. The assumptions are clearly stated in the relevant sections and the statements for the theorem and proposition.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The methodology presented in this paper should be simple enough to facilitate the reproduction of our results without any difficulty. That said, the reference implementation of our methodology is available at the following link: https://github.com/uoftcprg/pigames.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The reference implementation of our methodology is available at the following link: https://github.com/uoftcprg/pigames.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The parameters used in our experiments are clearly stated in the experimental section. The reference implementation of our methodology is available at the following link: https://github.com/uoftcprg/pigames.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The Elo differences between chess engines and their watermarked counterparts are presented with the standard information for chess engine performance, including the result's significance. Note that we invert the 'Likelihood of superiority' metric, reported by UCI chess tournament tool Cute Chess, to 'Likelihood of inferiority' to better match our use case, since we do not expect the watermarked engines to perform better than non-watermarked engines.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute resources are described in Appendix C. Note that our experiments are very lightweight and can easily be run on most modern computers.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our work does not involve human subjects, and we do not use any third-party dataset. Our work also does not entail any negative societal impact or harmful consequences.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: There is no societal impact of our work worth mentioning.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our work poses no risk of such a nature.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Our work uses numerous game opening books and engines, which we explicitly mentioned and whose licenses we complied with (if applicable). asmFish, Etheral, PlentyChess, Stash, Stockfish, GNU Go, Rapfi, and Fuego (opening books) are distributed under the GNU GPLv3 license. Dragon is proprietary freeware. KataGo is distributed under

the MIT license. Stockfish opening books are distributed under the Creative Commons Zero v1.0 Universal license. No license is specified for Gomocup 2025 opening books, but they are freely and publicly available.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We released a reference implementation, which is documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our work does not involve crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Our work does not involve the use of LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.