arXiv:2506.20664v1 [cs.AI] 25 Jun 2025

The Decrypto Benchmark for Multi-Agent Reasoning and Theory of Mind

Andrei Lupu[1,2], **Timon Willi**[1,2], **Jakob Foerster**[1,2]

[1]FAIR at Meta, [2]University of Oxford

As Large Language Models (LLMs) gain agentic abilities, they will have to navigate complex multi-agent scenarios, interacting with human users and other agents in cooperative and competitive settings. This will require new reasoning skills, chief amongst them being *theory of mind* (ToM), or the ability to reason about the "mental" states of other agents. However, ToM and other multi-agent abilities in LLMs are poorly understood, since existing benchmarks suffer from narrow scope, data leakage, saturation, and lack of interactivity. We thus propose Decrypto, a game-based benchmark for multi-agent reasoning and ToM drawing inspiration from cognitive science, computational pragmatics and multi-agent reinforcement learning. It is designed to be as *easy* as possible in all other dimensions, eliminating confounding factors commonly found in other benchmarks. To our knowledge, it is also the first platform for designing interactive ToM experiments.

We validate the benchmark design through comprehensive empirical evaluations of frontier LLMs, robustness studies, and human-AI cross-play experiments. We find that LLM game-playing abilities lag behind humans and simple word-embedding baselines. We then create variants of two classic cognitive science experiments within Decrypto to evaluate three key ToM abilities. Surprisingly, we find that state-of-the-art reasoning models are significantly *worse* at those tasks than their older counterparts. This demonstrates that Decrypto addresses a crucial gap in current reasoning and ToM evaluations, and paves the path towards better artificial agents.

Date: June 27, 2025
Correspondence: alupu@meta.com
Code and Data: https://github.com/facebookresearch/decrypto/

Meta

1 Introduction

Reasoning abilities and agentic behaviour are currently at the forefront of large language model (LLM) research (Huang et al., 2023; Jaech et al., 2024; Guo et al., 2025), with early applications of LLM agents including software engineering (Jimenez et al., 2023) and web navigation (Yao et al., 2023). While those tasks are predominantly single-agent, real-world agentic applications will increasingly involve complex interactions with many other actors, including business partners, teammates, users and other agents. Thus, the future of AI problems is irreducibly *multi-step*, *multi-agent*, *partially observable*, and *stochastic* – a reality not captured by the existing suite of benchmarks.

To navigate these multi-agent scenarios, LLMs will require *theory of mind* (ToM), the general ability of modelling the mental states of other agents (artificial or biological), including their knowledge, beliefs, intentions and abilities. However, LLM ToM remains poorly understood. Despite ToM being inherently tied to multi-agent *interactions*, the majority of existing benchmarks lack interactivity and are restricted to variants of the Sally-Anne task (Baron-Cohen et al., 1985). Furthermore, Hu et al. argue that many of those benchmarks suffer from confounding factors and bias introduced when translating embodied scenarios to text.

To address the gap in ToM evaluations, we introduce Decrypto, a multi-agent reasoning and ToM benchmark based on the award-winning language game of the same name. Decrypto is a pragmatic inference game (Degen, 2023), where two agents must exchange secret messages while preventing a third party from intercepting them.

We designed Decrypto to be future-proof and as *easy* as possible for LLMs in all other dimensions except

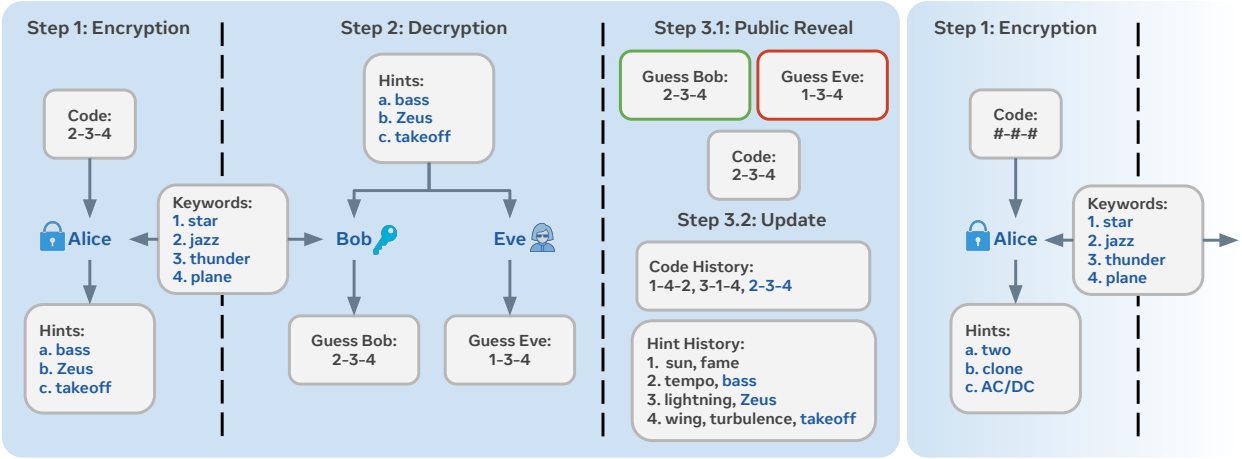

Figure 1 Left: Overview of a turn in DECRYPTO, split into three steps, with Alice and Bob (Encoder and Decoder) playing against Eve (Interceptor). Step 1: Alice gets a random code of 3 non-repeating digits, and provides 3 hints referring to the meaning of the four *Keywords*. Step 2: Bob and Eve receive the hints and attempt to guess the code independently. Step 3: Both guesses and the code are publicly revealed, and the code and hint histories are updated. All players have access to the histories at all times, but only Alice and Bob have access to the keywords. Alice aims to give hints that are easy for Bob to decode but hard for Eve. As the game progresses, the growing hint history makes it easier for Eve to intercept. Right: Encryption phase of a new turn. **We invite the reader to guess the code as Bob and then copy this for the answer: [ ].** We provide a rationale in Appendix C.

multi-agent reasoning. Indeed, it is operated entirely through word associations, eliminating common confounding factors of LLM performance, such as tokenisation, long contexts or embodied scenarios. The benchmark enables the study of cooperation, competition and ToM within a single game, and requires players to reason about the information available to others before making strategic decisions. Since it is based on a popular boardgame, our benchmark produces easily interpretable insights and enables the study of human-AI interactions in strategic communication tasks. Finally, to the best of our knowledge, DECRYPTO is the first platform for studying LLM ToM in interactive settings. Built to be easily extended, it enables researchers to create new ToM experiments by writing only prompts and a few lines of code. We demonstrate the versatility of the DECRYPTO ToM platform by creating variants of two classic experiments from cognitive psychology.

2 Decrypto: The Game

DECRYPTO is a word-based code-guessing board game published by Scorpion Masqué[1], of which we implement a variant requiring three players divided into two teams. Borrowing the terminology from cryptography, players are assigned three distinct roles: Alice (the Encoder), Bob (the Decoder), and Eve (the Interceptor or Eavesdropper). A full game (i.e., an episode) lasts at most 8 turns. We show one such turn in Figure 1.

Alice and Bob are on the same team, and share 4 ordered secret **keywords**, sampled without replacement before the start of the episode and kept fixed throughout. In Figure 1, those are `[star, jazz, thunder, plane]`.

Each turn, Alice samples a secret **code** of 3 non-repeating digits between 1 and 4, such as `[2-3-4]`, with any of the 24 possible codes occurring at most once per episode. Alice must then provide 3 public **hints** (e.g. `[bass, Zeus, takeoff]`, such that Bob can guess the code but Eve cannot.

Next, Bob and Eve each make an independent attempt to **guess** the code. If Bob guesses incorrectly, his team gets a **Miscommunication** token. If Eve guesses correctly, she gets an **Intercept** token.

Finally, Bob and Eve's guesses, the real code, and any event (miscommunication or intercept) that occurred during the turn are publicly revealed. Thus, the history of all codes from previous turns and the history of past hints associated with each of the four keywords are common knowledge, but the keywords themselves are

[1] www.scorpionmasque.com/en/decrypto

not. The **code history** and especially the **hint history** are the two main pieces of information that will help Eve make educated guesses in future turns. For instance, if the hint history for digit 3 so far is [lightning, Zeus], Eve may associate terms related to storms or electricity with that digit, even without knowing the exact keyword.

The game ends and Eve wins immediately if Alice and Bob accumulate two Miscommunication tokens or if Eve gains two Intercept tokens. Alice and Bob win if they make it through 8 rounds without either of those two conditions happening. Alice must therefore be careful to provide subtle hints to avoid interception, yet make them clear enough for Bob to guess correctly.

The keywords are sampled from a predetermined set, but *the choice of hints is open-ended* and can be composed of multiple words. The restriction is that hints must be real words (including proper nouns) and refer to the meaning of the keywords, not to their spelling or pronunciation. They must also rely on publicly available information, which includes culture, technical knowledge or recent events.

Beyond word associations

At the surface level, Decrypto provides a language reasoning challenge that consists in matching hints to either keywords or the hint history. However, Alice's hints cannot be too literal, or they will get intercepted, and so Decrypto can be formalised as a pragmatic inference game under the Rational Speech Act (RSA) framework (Goodman and Frank, 2016; Degen, 2023), where the listeners (Bob and Eve) update their belief of the intended meaning of the speaker (Alice) via Bayesian inference. We provide such a treatment in Appendix H, explicitly showing that agents must model each other's beliefs for optimal play. This includes the result that Bob must perform second-order ToM, modelling Alice's beliefs over Eve's beliefs, to maximize the chance of guessing correctly.

To help readers intuit the role of ToM in Decrypto, we also invite them to attempt guessing the code on the right side of Figure 1 as Bob, and to read the rationale in Appendix C. Alternatively, readers can download our code and play a game of Decrypto with an LLM in only a few minutes.

3 Decrypto: The Benchmark

An easy benchmark. Unlike other reasoning benchmarks, Decrypto is purposely designed to *not* require many of the capabilities that large pre-trained models struggle with. In particular, it demands no symbolic reasoning (Bard et al., 2020), mathematical reasoning (Cobbe et al., 2021; Zhang et al., 2024a), spatial reasoning (Clark et al., 2018; Carroll et al., 2019), tool use (Xu et al., 2023a), or particular attention to tokenisation, which has been shown to affect arithmetic performance (Singh and Strouse, 2024). It also distinguishes itself from recent ToM benchmarks (Le et al., 2019; He et al., 2023; Kim et al., 2023; Zhou et al., 2023; Xu et al., 2024; Gandhi et al., 2024; Sclar et al., 2024) by not relying on state tracking (e.g. of object locations) or on textual representations of embodied settings, which Hu et al. argue introduce unwanted pragmatic artifacts and bias LLM answers.

Instead, strong performance in Decrypto relies on language-based reasoning and theory of mind. Since LLMs are trained to learn word co-occurrences and follow instructions, we expect Decrypto to be easy to play by LLMs, but hard to master. Our results and Appendix L show this to be the case: even Llama 3.1-8B understands the rules, but much stronger models still struggle to play well.

A future-proof benchmark. With its two-team, three-player dynamics, Decrypto is a benchmark for both competitive and cooperative scenarios.

The difficulty scales with the choice of agents for each of the three roles, making the benchmark much harder to saturate than those that rely on a fixed dataset of problems. We curate a set of 680 possible keywords, resulting in over 8.8 billion possible keyword combinations to limit memorisation, and also implement the option to provide custom keywords beyond this standard set. The number of possible games is virtually infinite since the choice of hints is open-ended, and can rely on any publicly available information, including history, culture or science. Finally, the game was originally designed for human play, facilitating studies on human-AI coordination and comparisons.

3.1 Multi-Agent Reasoning

Competition. In this setting, we compare two models A and B by having both Alice and Bob be played by A, and Eve by B. At its simplest, we can record the win rate of A vs. B, swapping teams to ensure a fair evaluation. However, we find win-rate to be uninformative given the current abilities of LLMs, and propose different metrics below.

Ad-hoc Coordination. In this setting, we freeze Eve (e.g., to a rule-based baseline or the strongest available LLM) and instantiate Alice and Bob to be played by different models (i.e. Alice is A, and Bob is B). Like in Stone et al. (2010), we are concerned with the ability to "*efficiently and robustly collaborate with previously unknown teammates*", such as independently trained LLMs.

A crucial subset of ad-hoc coordination is **human-AI coordination**, where one of the two agents (Alice or Bob) is played by a human. This setting paves the way towards more social AI agents that seamlessly coordinate with humans and understand their intents.

Metrics. Both settings are subject to the same tension that is at the core of Decrypto: Alice must provide hints that balance what she knows about Bob, Eve, and the information available to each of them. If the hints are too obscure, Bob will guess wrong, which leads to a miscommunication; too obvious, and Eve will intercept; just right, and Alice and Bob survive for another round. The number of miscommunications and of intercepts are therefore two sides of the same coin, providing granular insights into the failure modes of LLMs. Meanwhile, the average number of turns per episode captures both sides in one metric, since longer games mean that Alice and Bob could better balance the difficulty of hints to avoid defeat. Game length is also more informative than win-rate, since we empirically find current LLMs to be much weaker at providing hints than intercepting, which results in Eve having significantly higher win-rates in most match-ups (see Figure 7).

3.2 Theory of Mind

As explained in Section 2, high performance in Decrypto requires players to explicitly reason about other players' beliefs and world knowledge. For example, sufficiently powerful LLMs playing the game could leverage their encyclopedic knowledge to gain an advantage over human players. Newer models could also win against older ones by referring to information made public after the knowledge cut-off of the opponent. For instance, a politician's name could be used to refer to the keyword "plane" based on recent news stories.

Beyond game playing, Decrypto provides a platform for conducting interactive ToM experiments inspired by cognitive psychology. We conduct two such experiments, evaluating three different ToM abilities in LLMs. Each required only new prompts and approximately 15 additional lines of code.

The first experiment adapts the Smarties Task of Gopnik and Astington, which presents children with a deceptive object (a box of Smarties containing pencils) and studies whether the child can correctly identify incorrect beliefs – either their own or of another child – when first encountering the deceptive object. To recreate this task in Decrypto, we substitute the closed Smarties box and the pencils with the game history and the secret keywords. At each turn except the first, we prompt Eve three times independently. Prompt A asks her to predict the four keywords. Prompt B reveals the keywords and asks Eve what she thought were the keywords pre-reveal. Prompt C also reveals the keywords but asks the model to predict what a "second interceptor" would think the keywords to be, pre-reveal.

We generate outputs with temperature 0 (for models that allow it) and only consider turns where the answer to A is an incorrect guess, keeping only cases where Eve has inaccurate "beliefs" pre-reveal. We compare answers A and B to measure **representational change** (RC), the ability of the agent to recognise when its belief about the world (but not the world itself) changes due to additional information. Similarly, comparing A and C measures **false belief** (FB), the ability to represent other agents as having false beliefs about the world. We distinguish two variants of the tasks. The *Weak* variant only requires the agents to realise that either themselves or the second interceptor could not have known the ground truth, and so an answer to either B or C is correct as long as it differs from the real keywords. For the *Strong* variant of those tasks, we consider the agent to pass only if it correctly predicts its answer to prompt A (i.e. if $B = A$ or $C = A$). Success here likely requires a self-consistent representation of the keywords, or at least strong counterfactual reasoning. Figure 5 summarises the procedure.

The second experiment measures **perspective taking** (PT) and is a word-based code-guessing variant of the Three Mountain Problem (Piaget et al., 1956). After Alice has given her hints, we prompt her to predict Eve's guess. We record the prediction accuracy and the proportion of turns on which Alice predicts that Eve will guess the correct code. This last metric is particularly informative: Alice predicting that Eve will intercept more rarely than she does is a sign that Alice fails to consider all information available to Eve. If, instead, Alice predicts that Eve will intercept on nearly every turn, it shows a failure to reason from Eve's perspective, and either explicitly or implicitly assumes that Eve has access to the keywords. It also shows a failure to integrate ToM reasoning in decision-making: if Alice thinks Eve will guess the code given her hints, she should have chosen different hints.

4 State of the Art

Given the plurality of settings in Decrypto, we caution users to be explicit and transparent about their assumptions when declaring state-of-the-art.

4.1 Specialist vs Generalist Agents

When reporting results on Decrypto, we recommend distinguishing between two classes of agents: **generalists** and **specialists**.

Generalist agents refer to general-purpose models used to play the game "out-of-the-box", without any additional fine-tuning or in-context learning outside of that happening within individual games. In particular, for generalist agents, we *consider the prompt to be the observation* returned by the environment, which precludes any form of prompt engineering. Evaluating foundation models in this framework assesses their ability to zero-shot generalise to novel tasks, including out-of-distribution ones and tasks where training data is limited or unavailable.

Because foundation models are often sensitive to prompting, we recommend evaluating generalist agents with a range of different prompts to prevent the choice of prompt favouring any one model. We provide such an evaluation using 625 prompt variants in Appendix G, showing that game performance is robust to prompt variations and depends predominantly on general model capabilities.

In contrast, **specialist agents** are all the agents designed specifically for the task. Solutions in this space include rule-based strategies, training on the game using RL, building a prompt pipeline informed by the rules of the games, and prompt engineering. Here, we *consider the prompt to be part of the agent*, if it is needed at all. Prompt engineering is the designer's burden, and the environment only returns key information.

Specialist agents can, in theory, overfit to Decrypto, which makes some degree of performance easier to reach, but does *not* automatically solve the game under every setting. For instance, we know from the cooperative multi-agent RL literature that training models to collaborate in cross-play (XP) with unseen agents can be extremely challenging, even with unlimited access to the environment (Hu et al., 2021; Cui et al., 2023). We expect this to also hold for Decrypto. This is evidenced by the rule-based baselines we describe next, which coordinate flawlessly in Decrypto if Alice and Bob pre-agree on a common strategy and word embeddings, but otherwise can fail catastrophically when paired with other agents, including humans and LLMs.

Specialist Agents - Baselines. We introduce two specialist agent baselines to play Decrypto. For the baseline models, we use the common crawl GloVe with a vocabulary of 2.2 million words (Pennington et al., 2014) and the Word2Vec embeddings trained on the Google News dataset, provided by the gensim library (Rehurek and Sojka, 2011; Mikolov, 2013).

For both embedding models, the baseline algorithm is the same and relies on a fixed corpus of 5696 nouns. When hinting a digit, Alice draws from the set of the top-K most similar words to the corresponding keyword under cosine similarity, filtering out hints that are more similar to another keyword than to the one intended. In turn, Bob greedily assigns each hint to the most similar keyword, measured using his own embedding model and cosine similarity. Not having access to the keywords, Eve computes the mean embedding of the hint history for each keyword, and then finds the assignment between hints and mean embedding that maximises

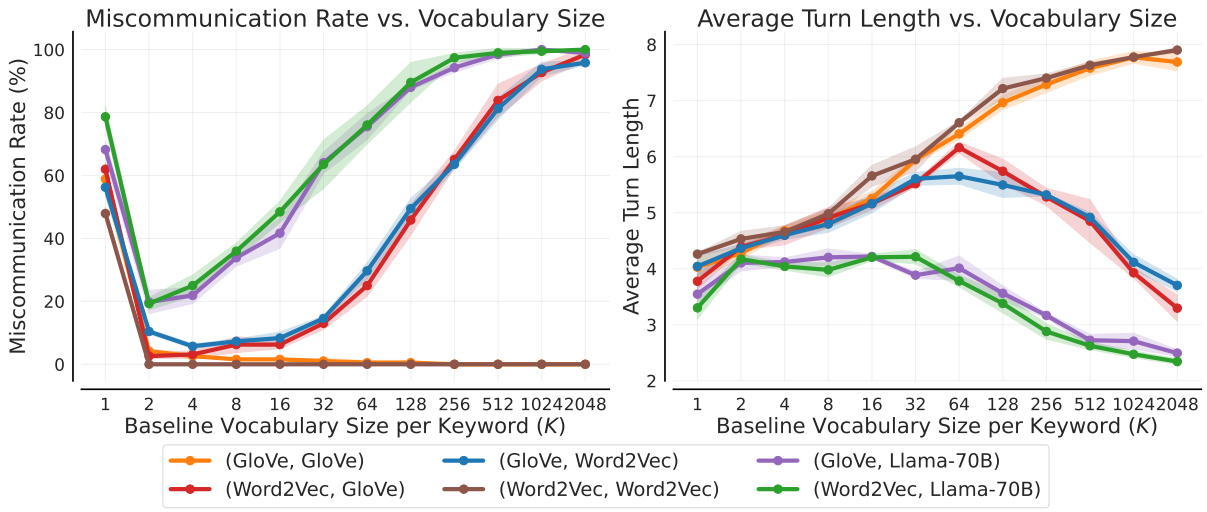

Figure 2 Percentage of games ending in miscommunications (left) and average game length (right) for the word embedding baselines as a function of K – the hint vocabulary size per keyword. Each curve corresponds to an (encoder, decoder) pair. For low K, the baselines coordinate well, but their hints are easier to intercept. At high K values, both models sample from lower similarity words. This results in very strong self-play baselines, which cannot coordinate in cross-play, with most games ending in miscommunication for $K > 128$. While the interceptor is Llama 3.1-70B, trends hold across all interceptors, including baselines. We report mean and standard error over 3 model seeds.

the sum of cosine similarities between each hint and the history of the digit it is assigned to. We provide additional algorithmic details in Appendix E.

The purpose of these baselines is threefold. First, we can study the case where Alice and Bob share perfect ToM, represented by a shared strategy and word embedding. This establishes an upper bound on performance to which to compare other agents. Second, having baselines using two different word embeddings, we demonstrate in Section 5 how not accounting for differences in how agents associate words can lead to a catastrophic lack of coordination and poor XP performance. Third, K determines the amount of diversity in Alice's hints. By tweaking it, we can operate in a regime where GloVe and Word2Vec correlate and represent reasonable rule-based baselines against which we can compare LLMs. We thus use $K = 16$ for the rest of the paper.

Being explicit about whether Decrypto is approached from a *generalist* or *specialist* perspective provides clarity and avoids unfair comparisons between models making inherently different assumptions. It also leads to two equally rich but distinct research directions. In the first case, Decrypto acts as a benchmark for multi-agent reasoning and measures progress in foundation models. In the second case, it is an RL environment used to research and evaluate training algorithms.

4.2 Human-Data Collection

We collect 10 full games of human teams playing against a fixed LLM Eve, namely Llama 3.1-70B-Instruct. The players interacted with the game through a unified command-line interface and saw the same system prompts, user prompts and information as LLMs did. See Appendices I and J for prompts and Appendix K for data collection details. The data itself is hosted in the code repository.

Normally, a game ends when 2 interceptions or miscommunication tokens are collected. To maximise our data collection, we did not terminate the game after 2 interceptions but let the humans play the game for the full 8 turns. The participants were encouraged to keep providing good hints and analyse the hints appropriately, even after the game technically terminated. Such a game does not count as won, but the extra turns can be useful when replaying the game to evaluate other LLMs as Eve.

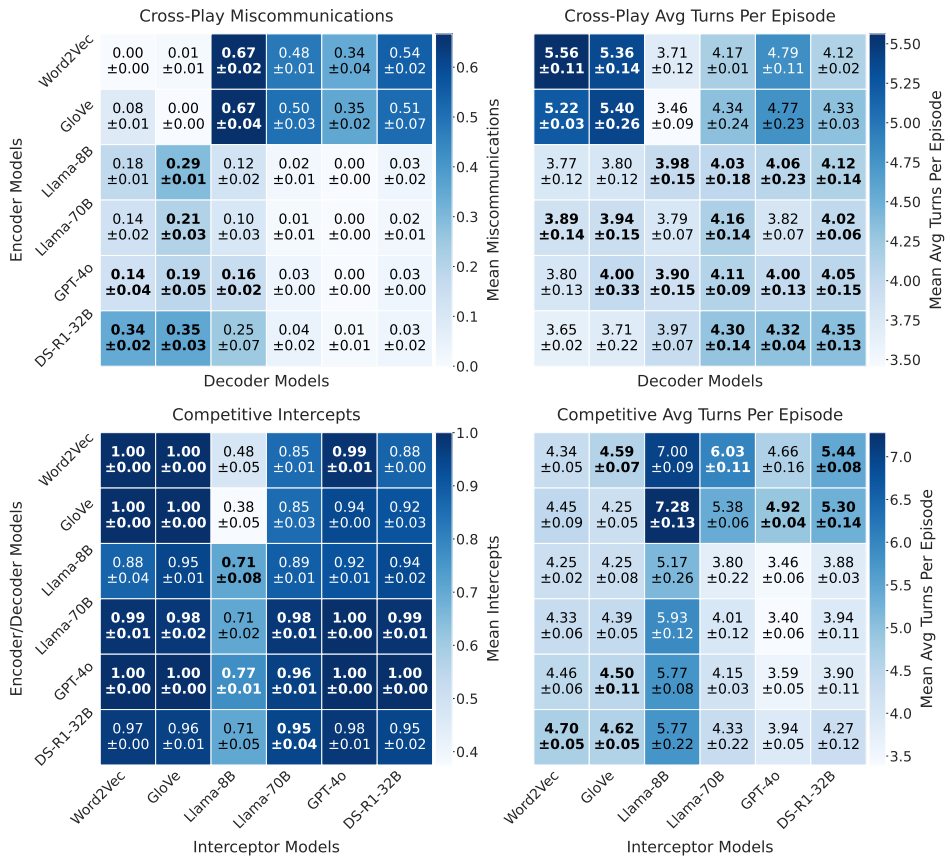

Figure 3 Cooperative and competitive results for 6 different agents. "Llama" refers to "Llama 3.1-Instruct"; "DS-R1-32B" is "DeepSeek-R1-Distill-Qwen-32B". All results are reported for 32 games and 3 model seeds. **Top:** Cooperative cross-play matrix when the interceptor is Llama 3.1-70B-Instruct. *Left:* proportion of games ending in miscommunication. *Right:* average number of turns per episode. At K=16, baselines play well with each other, outperforming all LLM agents tested, but coordinate poorly with LLMs. **Bottom** Competitive results playing a homogeneous encoder-decoder team against an interceptor. *Left:* proportion of games ending in an intercept. *Right:* average number of turns per episode. Baselines outperform LLM agents across the board. DeepSeek-R1-Distill-Qwen-32B is the strongest LLM when looking exclusively at inter-LLM coordination and competition, but is less compatible with baselines and weaker at intercepting them than GPT-4o.

5 Results

To kickstart this benchmark, we assess the cooperative, competitive, and theory of mind capabilities of specialised and generalist agents in Decrypto. Amongst generalist open-source models, we evaluate Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct (Dubey et al., 2024), and DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025). From the closed-source models, we only evaluate GPT-4o due to resource constraints. We also include the two baselines mentioned in Section 4.1. Finally, we present human-AI coordination and competition results. Experimental details can be found in Appendix F. Unless otherwise mentioned, we report mean $\pm$ standard error over three seeds for all experiments.

Baselines. We first look at the specialised agents baselines and show in Figure 2 that we can control K for the top-K selection to make them arbitrarily strong. Smaller K prioritises words semantically similar to the keyword, making the hints easier to guess. Bigger K selects semantically less similar hints, which are harder to guess if Bob cannot access the same embeddings. For example, suppose GloVe-Alice wants to hint the keyword "fire" to Word2Vec-Bob, and $K = 50$. The 50th most similar word to "fire" from the hint corpus would be "oil" for GloVe. However, "oil" is not even in the top 1000 most similar words for Word2Vec-Bob, highlighting a situation where cross-play difficulties arise. This finding also holds when an LLM plays Bob. As K increases, miscommunications increase, and the average turn length decreases.

Model as Interceptor	Miscomms	Intercepts(↑)	Win Rate(↓)	Game Length(↓)
Word2Vec	11.00 ± 0.00	8.00 ± 0.00	40.00% ± 0.00%	7.00 ± 0.00
GloVe	9.00 ± 0.00	9.00 ± 0.00	**20.00% ± 0.00%**	6.50 ± 0.00
Llama 3.1-8B-Instruct	11.67 ± 0.33	6.33 ± 0.88	46.67% ± 3.33%	6.97 ± 0.13
ExploreToM-8B	11.67 ± 0.33	4.00 ± 2.31	43.33% ± 6.67%	6.90 ± 0.20
Llama 3.1-70B-Instruct	11.33 ± 0.33	8.33 ± 1.76	46.67% ± 3.33%	6.80 ± 0.21
GPT-4o	10.67 ± 0.88	10.00 ± 0.58	36.67% ± 3.33%	**6.50 ± 0.15**
DeepSeek-R1-Dist-32B	9.33 ± 0.33	10.67 ± 0.88	30.00% ± 5.77%	**6.33 ± 0.03**
DeepSeek-R1	9.33 ± 0.33	**15.00 ± 0.00**	**20.00% ± 5.77%**	**6.27 ± 0.12**
Claude 3.7 (+ ext. thinking)	8.33 ± 0.33	14.00 ± 0.58	**20.00% ± 0.00%**	**6.30 ± 0.12**
Model as Decoder	**Miscomms(↓)**	**Intercepts**	**Win Rate(↑)**	**Game Length(↑)**
Word2Vec	20.00 ± 0.00	3.00 ± 0.00	0.00% ± 0.00%	2.70 ± 0.00
GloVe	20.00 ± 0.00	5.00 ± 0.00	0.00% ± 0.00%	3.10 ± 0.00
Llama 3.1-8B-Instruct	20.00 ± 0.00	4.33 ± 0.88	0.00% ± 0.00%	3.00 ± 0.15
ExploreToM-8B	20.00 ± 0.00	3.00 ± 0.00	0.00% ± 0.00%	2.53 ± 0.09
Llama 3.1-70B-Instruct	18.67 ± 0.88	7.67 ± 0.33	3.33% ± 3.33%	4.70 ± 0.12
GPT-4o	17.33 ± 0.88	8.67 ± 0.67	3.33% ± 3.33%	5.23 ± 0.12
DeepSeek-R1-Dist-32B	19.67 ± 0.33	6.67 ± 1.20	0.00% ± 0.00%	4.43 ± 0.22
DeepSeek-R1	18.67 ± 0.88	7.33 ± 0.33	6.67% ± 6.67%	5.53 ± 0.13
Claude 3.7 (+ ext. thinking)	12.67 ± 0.67	11.00 ± 0.58	16.67% ± 6.67%	6.57 ± 0.12
Human (original games)	**11**	12	**40.00%**	**6.90**

Table 1 We collect 10 games from human encoder-decoder teams and report the agents' performance when playing as (top) interceptor against human players or as (bottom) decoder with a human encoder. Claude 3.7 with extended thinking is the strongest model in both roles, although that does not translate to shorter games when it acts as an interceptor. ExploreToM-8B, a Llama 3.1-8B variant fine-tuned by Sclar et al. on ToM data, performs worse than the latter in both settings. In ad-hoc human coordination, all agents are significantly worse than humans, getting more miscommunications, a lower win rate and shorter games. We also see DeepSeek-R1-Dist-32B breaking the trend from Figure 3 by performing worse than GPT-4o and Llama 3.1-70B in this setting.

Coordination. Baseline-LLM teams get significantly more miscommunications than baseline-baseline or LLM-LLM teams. The top row of Figure 3 shows the total number of games ending in miscommunication out of 32 games for the two baselines and two LLMs, Llama-3.1-8B and Llama-3.1-70B. Among LLMs, the main determining factor for miscommunications is the model used for Bob, with the smallest model, LLama-3.1-8B, seeing the most miscommunications. Interestingly, we do not observe any self-play/cross-play gap, even though it would technically be possible for an LLM to perfectly model its counterparts when playing with or against the same model.

Competition. For our competitive results, larger models generally perform better as both Alice/Bob and Eve. However, we find that the win rate is heavily skewed in favour of Eve, as shown in Figure 7, with most models rarely surviving. We measure the number of interceptions and average game length of different SP teams against different Eve Agents. We report our results in the bottom row of Figure 3. However, as our human experiments show, such a heavy bias is not a property of the game but of the LLMs themselves. Indeed, we find that novice humans achieve 33% win rate against even the strongest Eve agents.

Human Evaluation. Table 1 shows the result of replaying collected human games by re-using the human-provided hints and replacing either Eve or Bob with different LLMs. When Alice and Bob are humans, we find the strongest interceptors to be DeepSeek-R1 and Claude 3.7, with the most intercepts and the lowest win rate. We next fix Eve's guesses to match the data and replace Bob with different LLMs. Surprisingly, DeepSeek-R1-32B, which showed strong results in Figure 3, trails significantly behind both LLama-3.1-70B and GPT-4o. More importantly, all decoders tested (LLMs or baselines) fall short of the human decoders, with only Claude 3.7 with extended thinking coming close. This demonstrates a significant gap for most models in their ability to coordinate with humans.

Theory of Mind. We show results for both experiments in Figure 4. Most models tested achieve high accuracy in both weak RC and weak FB, with Llama 3.1-70B slightly outperforming newer models, including Claude 3.7 with extended thinking and o1 high. In contrast, all models get $\leq 10\%$ accuracy on the strong tasks,

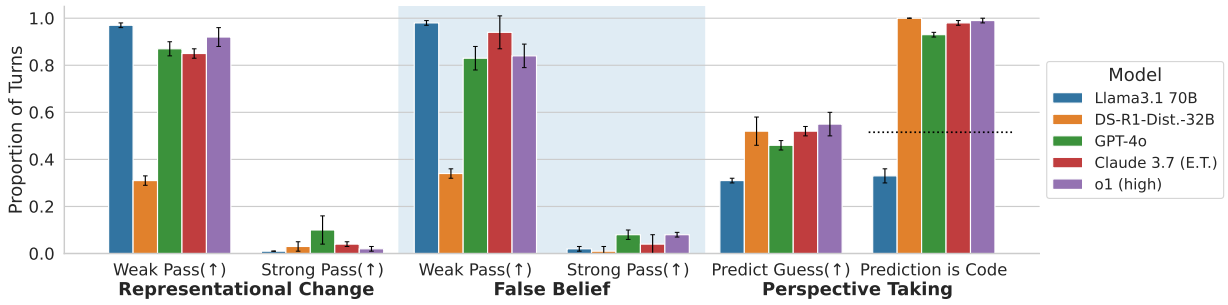

Figure 4 *Representational change* (RC), *false belief* (FB) and *perspective taking* (PT) scores in Decrypto. RC measures whether agents can identify a change in their own world-view following new information. FB measures whether agents can model the incorrect beliefs of another agent. PT assesses whether agents can reason from the point of view of an agent with limited information. All LLMs evaluated struggle with PT and with the strong variants of RC and FB. Surprisingly, Llama 3.1-70B outperforms recent reasoning models in all three ToM tasks. In PT, models regularly fail to consider the other agent's point of view, instead answering questions based on privileged information only accessible to them. We discuss experimental details and further insights in the main text.

showing poor self-consistency even when generating outputs with 0 temperature. In PT, we see Llama trail behind other models in prediction accuracy. However, the last metric in Figure 4 reveals that all models except Llama predict that Eve will intercept on nearly every turn, when the real interception rate is $\sim 52\%$, indicated by the dotted line. Inspecting model outputs in Appendix M, we find those models to predict that Eve will intercept *even on the first turn*, with only Llama correctly pointing out that Eve can do no better than a random guess. Surprisingly, Listing 16 shows these results to hold even if we modify the PT prompt to emphasize that Eve "*does *NOT* know the secret keywords*". As stated above, this is a double failure of ToM: first, as a failure to reason from Eve's perspective, and second, because the model failed to consider how the hints will be received when choosing them.

6 Related Work

BattleAgentBench (Wang et al., 2024), multi-robot studies (Li et al., 2023) and other works such as Liu et al. (2023) and Guo et al. (2024) have recently studied LLMs in multi-agent contexts. While detailed, they focus on spatial reasoning rather than on the language-based reasoning and ToM abilities central to Decrypto. Among multi-agent environments, social deduction games such as Werewolf (Xu et al., 2023b,c) and Avalon (Light et al., 2023) are particularly amenable to studying LLMs, as are negotiation games (Abdelnabi et al., 2023). However, they provide less structured and atomic insight than Decrypto, given the free-form dialogue and frequent deception in those settings. AgentPro (Zhang et al., 2024b) focuses on competitive scenarios in games like Blackjack and Poker, but lacks the cooperative aspect of Decrypto.

ToM has been a subject of focus in AI as early as 2018, when Rabinowitz et al. trained ToM-net, a simple network capable of predicting the behaviour of other artificial agents and acting based on false beliefs. In contrast, recent works on ToM have focused on LLMs modelling human beliefs (Chen et al., 2024; Sclar et al., 2024; Kim et al., 2023). Decrypto connects both by enabling the study of LLM ToM applied to both humans and other artificial agents. Unlike other recent ToM benchmarks (Le et al., 2019; He et al., 2023; Kim et al., 2023; Zhou et al., 2023; Xu et al., 2024; Gandhi et al., 2024; Sclar et al., 2024) and Rabinowitz et al. (2018), however, Decrypto does so in an interactive setting, where agents can not only model the beliefs of other agents, but also influence them.

Kosinski (2023) also studied the Smarties Task, from which they concluded that "ToM-like ability [...] may have spontaneously emerged" in LLMs. Those claims were directly challenged by Ullman in subsequent work, with which our results agree. Shaikh et al. (2023) use another language game, Codenames, to study cross-cultural pragmatics. Their work stops short of evaluating LLMs, but raises important questions about the role of cultural backgrounds in pragmatic inference. Decrypto is a natural choice for pursuing research in this direction, since we can specify different personas in the LLM prompts.

7 Conclusion

Our paper introduces DECRYPTO, an interactive multi-agent benchmark designed to evaluate coordination, competition and theory of mind capabilities in language models. DECRYPTO addresses critical limitations of existing ToM benchmarks, such as biases arising from textual translation of embodied scenarios or lack of interactivity. Designed to be future-proof and to eliminate confounding factors known to limit LLM performance, DECRYPTO fills an important gap in existing benchmarks. Furthermore, our codebase provides a versatile platform for quickly designing interactive ToM experiments inspired by cognitive psychology.

We conduct extensive experiments to evaluate open-source and closed-source LLMs. We find that even state-of-the-art models struggle with the nuanced communication and strategic reasoning that DECRYPTO requires, often underperforming simple baselines in cooperative and competitive settings. Similarly, our human-AI experiments shed light on the limited ability of recent LLMs to coordinate with humans or understand their communications.

Finally, we create variants of two seminal experiments from cognitive psychology to study LLM ToM in interactive settings, analogously to how it is studied in children. Our experiments provide strong evidence that state-of-the-art still lack many ToM skills. Moreover, we find newer and more capable reasoning models such as Claude 3.7 Sonnet and o1 high to be significantly worse at some ToM tasks than older models, demonstrating the need for more ToM methods and benchmarks.

DECRYPTO opens up several promising research directions. On the ToM side, these include enhancing the ToM capabilities of language models and investigating differences between how LLMs model humans versus how they model other agents. From the multi-agent perspective, an exciting avenue for future work is adapting algorithms from multi-agent RL (Silver et al., 2017; Hu et al., 2020) to fine-tune LLMs. For a longer discussion on the limitations of our work and on the multiple research directions enabled by DECRYPTO, we refer the reader to Appendix A.

Acknowledgements

We extend our gratitude to Roberta Raileanu for early discussions on this project and to Prof. Alison Gopnik for advice on our theory of mind experiments. We also thank Mattie Fellows for assistance with the theory surrounding the RSA formalism.

AL is partially funded by the FRQNT doctoral training scholarship (doi.org/10.69777/321287). JF is partially funded by the UKRI grant EP/Y028481/1 (originally selected for funding by the ERC).

References

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. Llm-deliberation: Evaluating llms with interactive multi-agent negotiation games. *arXiv preprint arXiv:2309.17234*, 2023.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

Simon Baron-Cohen, Alan M Leslie, and Uta Frith. Does the autistic child have a "theory of mind"? *Cognition*, 21(1): 37–46, 1985.

Leon Bergen, Roger Levy, and Noah Goodman. Pragmatic reasoning through semantic inference. *Semantics and Pragmatics*, 9:20–1, 2016.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Zhuang Chen, Jincenzi Wu, Jinfeng Zhou, Bosi Wen, Guanqun Bi, Gongyao Jiang, Yaru Cao, Mengting Hu, Yunghwei Lai, Zexuan Xiong, et al. Tombench: Benchmarking theory of mind in large language models. *arXiv preprint arXiv:2402.15052*, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Brandon Cui, Andrei Lupu, Samuel Sokota, Hengyuan Hu, David J Wu, and Jakob Nicolaus Foerster. Adversarial diversity in hanabi. In *The Eleventh International Conference on Learning Representations*, 2023.

Judith Degen. The rational speech act framework. *Annual Review of Linguistics*, 9(1):519–540, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

Michael C Frank and Noah D Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084): 998–998, 2012.

Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah Goodman. Understanding social reasoning in language models with language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination. *arXiv preprint arXiv:2503.17821*, 2025.

Noah D Goodman and Michael C Frank. Pragmatic language interpretation as probabilistic inference. *Trends in cognitive sciences*, 20(11):818–829, 2016.

Alison Gopnik and Janet W Astington. Children's understanding of representational change and its relation to the understanding of false belief and the appearance-reality distinction. *Child development*, pages 26–37, 1988.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L Griffiths, and Mengdi Wang. Embodied llm agents learn to cooperate in organized teams. *arXiv preprint arXiv:2403.12482*, 2024.

Yinghui He, Yufan Wu, Yilin Jia, Rada Mihalcea, Yulong Chen, and Naihao Deng. Hi-tom: A benchmark for evaluating higher-order theory of mind reasoning in large language models. *arXiv preprint arXiv:2310.16755*, 2023.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *International Conference on Machine Learning*, pages 4369–4379. PMLR, 2021.

Jennifer Hu, Felix Sosa, and Tomer Ullman. Re-evaluating theory of mind evaluation in large language models. *arXiv preprint arXiv:2502.21098*, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Justine T Kao, Jean Y Wu, Leon Bergen, and Noah D Goodman. Nonliteral understanding of number words. *Proceedings of the National Academy of Sciences*, 111(33):12002–12007, 2014.

Hyunwoo Kim, Melanie Sclar, Xuhui Zhou, Ronan Le Bras, Gunhee Kim, Yejin Choi, and Maarten Sap. Fantom: A benchmark for stress-testing machine theory of mind in interactions. *arXiv preprint arXiv:2310.15421*, 2023.

Michal Kosinski. Theory of mind may have spontaneously emerged in large language models. *arXiv preprint arXiv:2302.02083*, 4:169, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Matthew Le, Y-Lan Boureau, and Maximilian Nickel. Revisiting the evaluation of theory of mind through question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5872–5877, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1598. https://aclanthology.org/D19-1598/.

Huao Li, Yu Quan Chong, Simon Stepputtis, Joseph Campbell, Dana Hughes, Michael Lewis, and Katia Sycara. Theory of mind for multi-agent collaboration via large language models. *arXiv preprint arXiv:2310.10701*, 2023.

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. Avalonbench: Evaluating llms playing the game of avalon. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

Jijia Liu, Chao Yu, Jiaxuan Gao, Yuqing Xie, Qingmin Liao, Yi Wu, and Yu Wang. Llm-powered hierarchical language agent for real-time human-ai coordination. *arXiv preprint arXiv:2312.15224*, 2023.

Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

Jean Piaget, Bärbel Inhelder, Frederick John Langdon, and JL Lunzer. *La Représentation de L'espace Chez L'enfant. The Child's Conception of Space... Translated... by FJ Langdon & JL Lunzer. With Illustrations.* New York; Routledge & Kegan Paul: London; printed in Great Britain, 1956.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.

Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

Melanie Sclar, Jane Yu, Maryam Fazel-Zarandi, Yulia Tsvetkov, Yonatan Bisk, Yejin Choi, and Asli Celikyilmaz. Explore theory of mind: Program-guided adversarial data generation for theory of mind reasoning. *arXiv preprint arXiv:2412.12175*, 2024.

Omar Shaikh, Caleb Ziems, William Held, Aryan J Pariani, Fred Morstatter, and Diyi Yang. Modeling cross-cultural pragmatic inference with codenames duet. *arXiv preprint arXiv:2306.02475*, 2023.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Aaditya K Singh and DJ Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *arXiv preprint arXiv:2402.14903*, 2024.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

Tomer Ullman. Large language models fail on trivial alterations to theory-of-mind tasks. *arXiv preprint arXiv:2302.08399*, 2023.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Wei Wang, Dan Zhang, Tao Feng, Boyan Wang, and Jie Tang. Battleagentbench: A benchmark for evaluating cooperation and competition capabilities of language models in multi-agent systems. *arXiv preprint arXiv:2408.15971*, 2024.

Hainiu Xu, Runcong Zhao, Lixing Zhu, Jinhua Du, and Yulan He. Opentom: A comprehensive benchmark for evaluating theory-of-mind reasoning capabilities of large language models. *arXiv preprint arXiv:2402.06044*, 2024.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models, 2023a.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*, 2023b.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940*, 2023c.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*, 2024a.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*, 2024b.

Pei Zhou, Aman Madaan, Srividya Pranavi Potharaju, Aditya Gupta, Kevin R McKee, Ari Holtzman, Jay Pujara, Xiang Ren, Swaroop Mishra, Aida Nematzadeh, et al. How far are large language models from agents with theory-of-mind? *arXiv preprint arXiv:2310.03051*, 2023.

A Limitations and Future Work

We firmly believe Decrypto to be a valuable and future-proof benchmark for evaluating multi-agent reasoning and theory of mind. However, our work still has a number of limitations, which we hope to address in future work.

First, while we largely eliminated the confounding factors that affect LLM performance, the game dynamics and the ToM tasks may still be too complicated for weaker models. For instance, when trying Llama 3.1-8B-Instruct on the perspective taking task, we found that it failed to understand the prompt altogether.

Regarding evaluations, the main limitation of our work is the computational cost of computing Figure 3. An individual run is relatively cheap (approximately \$0.10 per role per episode for Claude 3.7 Sonnet). However, given the 3-player nature of the game, the total number of combinations for n models is n^3, or $2n^2$ if only computing the matrix slices in Figure 3. For a large number of models, this can induce significantly larger costs than standard single-agent benchmarks. To mitigate this, we provide the tools to run evaluations locally and efficiently, leveraging prefix caching where possible to reduce unnecessary computations. We also acknowledge the limited scope of our human study in Table 1. While the 10 games collected were sufficient to demonstrate the limitations of LLMs in human-AI interactions, we believe that there is much to be learned from a more extensive study, including with humans as interceptors. We therefore made sure to provide the tools to enable such research, such as a lightweight command-line interface for conducting human studies.

Finally, theory of mind is an umbrella term for multiple abilities. Our work evaluates some of them, such as representational change, false belief and perspective taking, and we hope extensions to Decrypto to expand on this set in the future. Nevertheless, there are ToM abilities which Decrypto is ill-fitted to evaluate. For instance, models performing well in the game and the ToM tasks we designed may still fail to reason about the intentions or emotions of other agents.

Decrypto also enables a number of other research directions connected to ToM, multi-agent reasoning and reinforcement learning, and pragmatic inference. From a ToM perspective, Decrypto is easy to extend with additional ToM experiments, which can complement the results presented in our work. Given the limitations of current frontier models, we also believe there is a dire need for methods to improve the ToM abilities of LLMs. Decrypto features short episodes and dense rewards, making it a suitable environment to study the application of multi-agent RL algorithms towards LLM post-training. In particular, Decrypto features stochasticity and partial observability, two key ingredients for "meaningful" coordination challenges (Ellis et al., 2023). It would therefore be interesting to study whether fine-tuning LLMs on Decrypto using self-play (Silver et al., 2017) or other MARL algorithms (Hu et al., 2020) produces the same kind of idiosyncrasies observed when training *tabula rasa* models in other cooperative environments (Cui et al., 2023; Gessler et al., 2025). Finally, we believe the study of pragmatic inference in LLMs to be a promising avenue for future research, which is made much easier by the release of our benchmark. We formalise Decrypto as a pragmatic inference game and provide suggestions for this line of work in Appendix H.

B Impact Statement

As LLMs become more prevalent in our daily lives and start interacting with each other and with human users, we believe it becomes crucial to understand the multi-agent and ToM abilities of foundation models. Our work provides a novel assessment of those abilities, and the tools to conduct future studies on inter-AI and human-AI interactions. As such, we hope to bring attention to what we consider to be an understudied subset of agentic capabilities, and that our work will lead to models with a strong social and human-centric behaviour.

However, ToM encompasses a broad set of abilities, of which Decrypto evaluates only a subset. A possible misuse of Decrypto is to equate strong performance on our benchmark to more general ToM abilities, including intent inference and emotion understanding, which are not evaluated by Decrypto. Such claims have incorrectly been made about other ToM benchmarks (Hu et al., 2025), and so we ask future researchers to be careful when concluding that an AI system has ToM.

C Rationale for the code in Figure 1

The code that Alice encrypted in Figure 1 is `[4-1-3]`. The rationale is at follows. The four keywords are `[1. star, 2. jazz, 3. thunder, 4. plane]`, and the hints are `[two, clone, AC/DC]`. "Two" refers to "two dimensions", which is the defining characteristic of a geometric plane. "Clone" is a hint for "star", since both "clone" and "star" are common operations performed on a GitHub repository. "AC/DC" is a rock band, and one of their most famous songs, "Thunderstruck" starts with the word "Thunder" being chanted repeatedly.

The choice of hints relies on multiple assumptions made by Alice, here played by the authors, about Bob, played by the readers. The first hint relies on the assumption that many readers will have a mathematical background and quickly think of "plane" in its geometric sense, rather than as an aircraft. Similarly, we assume for the second hint that readers will be sufficiently familiar with GitHub to make the same connection as us. Failing that, we also anticipated that some readers might link the two terms by thinking of the "clone troopers" from Star Wars, which also leads to a correct guess. Finally, we assumed that the song "Thunderstruck" is famous enough to be picked up by most readers.

This is an example of how Alice can use ToM and the specific information she has about another player, in this case the readers, to inform her choice of hints. We hope we were successful.

D Diagrams for Theory of Mind Experiments

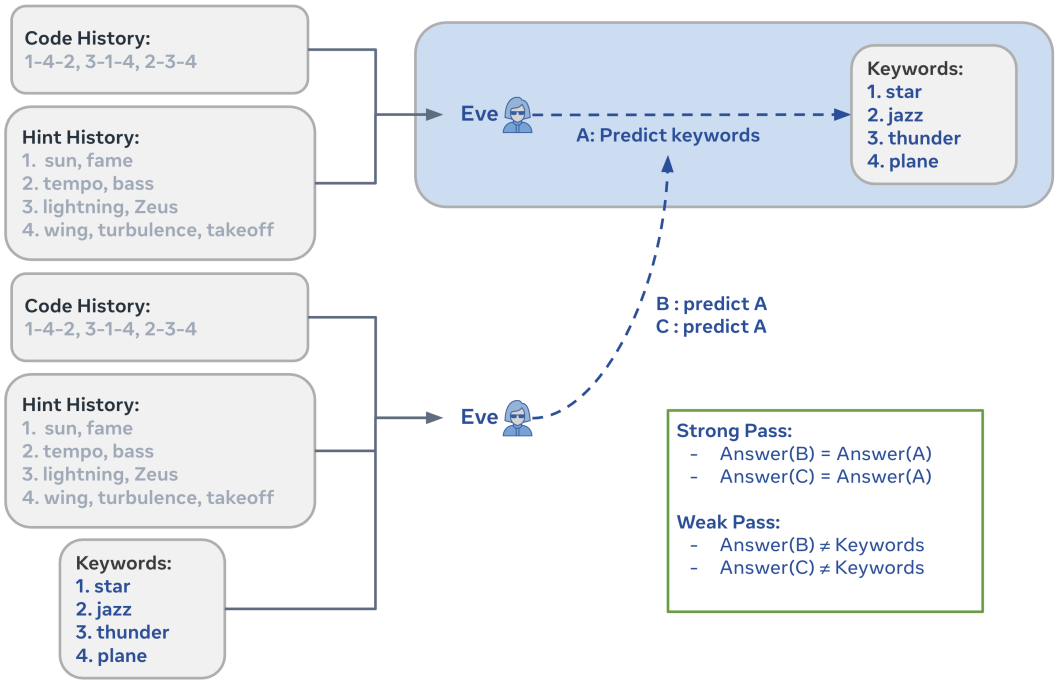

Figure 5 Diagram of the ToM experiment evaluating Representational Change and False Belief. Eve is prompted three times. A) She is prompted to predict the keywords given the code and hint histories. B) Eve is revealed the keywords and is asked what she thought were the four keywords pre-reveal. C) Eve is also revealed the keywords and is asked what a second interceptor would think are the keywords pre-reveal. For B or C, we consider it a strong pass if the answer matches the answer to A. We consider it a weak pass if he answer is *not* the keywords. Because we only evaluate on turns where Answer(A) $\neq$ Keywords, a strong pass always implies a weak pass as well.

E Baseline Details

Here we provide additional details on the word-embedding baselines introduced in Section 4.1.

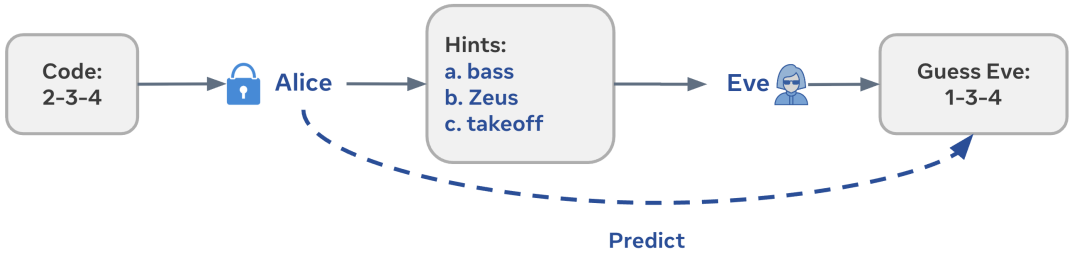

Figure 6 Diagram of the ToM experiment evaluating perspective taking. After Alice has given her hints, we prompt her to predict Eve's guess.

The baselines work as follows. Alice can choose hints from a fixed corpus of the 5696 most common nouns extracted from the Brown, Gutenberg, and Webtext corpora (Bird et al., 2009), filter to ensure all extracted nouns are present in both the GloVe and Word2Vec embeddings. (We use this hint corpus for the baselines only; LLM agents generate hints in an open-ended fashion.)

Before Alice chooses hints, Alice and Bob agree on a strategy. For each code digit $c_i \in [1, 4], i \in [1, 3]$, Alice picks a hint h_i and guarantees that the hint picked for the digit is more similar to the c_i-th keyword k_{c_i} than to any other keyword, i.e., $s(h_i, k_{c_i}) > s(h_i, k_j) \quad \forall j \in -\mathbf{c}$, where $-\mathbf{c} = [1, 4] \setminus c_i$ and s is the cosine similarity.

Alice increases the diversity of her hints by randomly sampling each hint from the top-K most similar words for each keyword under cosine similarity, filtering those that do not satisfy the constraint above. We also enforce that Alice cannot reuse any previously used hints in the same episode.

Bob then analyses the hints and assigns each hint to its most similar keyword under cosine similarity, i.e., $\arg\max_{c \in [1,4]} s(h_i, k_c) \ \forall i \in [1, 3]$. Bob guesses the code perfectly if Alice and Bob use the same word embeddings because the hints are guaranteed to be the most similar to the correct keyword under the same embedding model.

Eve is only given the hints $\mathbf{h}$ and the hint history for each keyword, which at turn 0 is empty. Eve calculates the cosine similarity between the average embedding of each keyword's hint history with the hints. This results in a similarity matrix of size $N \times M$, where $N(= 3)$ is the number of hints and $M(= 4)$ is the number of keywords. Eve combinatorially calculates the globally optimal guess based on the similarity values. Since N and M are typically small values, we are unconcerned about the computational complexity. However, Eve's selection task is equivalent to a linear assignment problem and efficient algorithms, such as the Jonker-Volgenant algorithm, are available in open-source libraries such as SciPy (Virtanen et al., 2020).

F Experimental Details

Here we provide details for the experiments reported in Section 5.

F.1 Prompting

All our experiments are text-only and conducted exclusively through prompts, which we provide in Appendices I and J. The system prompt is composed of an explanation of the game rules, common to all roles, and a set of role-specific instructions. The user prompts are role-specific but follow a similar structure, providing, in order, a summary of the previous turn and of the game so far, role specific information (e.g. the code or the hints), a summary of the role task, and formatting instructions. For the ToM tasks, the prompts are structure similarly, and rely on the same output formatting. For agents that do not admit a system prompt, such as Deepseek-R1, we include the system prompt as a prefix to the first user message.

We maintain prompts and model outputs in context, making Decrypto a multi-turn environment. We use a regular expression to extract the hints or the guess from the model outputs. If the code fails to detect a

correctly-formatted answer in the output, we re-prompt the model again up to 10 times, reminding it of the correct output format. If the model has still not provided an answer after 10 attempts, we default to a dummy answer. In practice, this almost never happens unless the output token limit is set very low for very verbose models, such as Deepseek-R1. By default, retries are not added to the model's context to avoid cluttering it and unnecessarily lowering its performance. ToM prompts and outputs are also excluded from the context for future turns.

F.2 Generation parameters

We set the model temperature to 0.6 for experiments in Figure 3 and table 1, and to 0 for ToM experiments in Figure 4 to reduce the impact of model stochasticity. We note that some close source models are not entirely deterministic, even with the temperature set to 0.

Our goal was not to evaluate model verbosity, but rather the limit of their capabilities. Therefore, we set generous token limits (between 750 for non-reasoning models and up to 10000 for reasoning ones) to prevent cutting model generations prematurely. However, we found that models rarely got close to their token limit. For instance, Claude 3.7 Sonnet with Extended Thinking output approximately 500 total tokens per turn on average, and sometimes as low as 350 despite a limit of 1500 tokens.

The full set of generation parameters can be found in the configs we provide as part of our code release, and which enable reproduction of the results in our paper with only a few commands.

F.3 Computational Costs

We ran our experiments on a mix of local and API-hosted models.

All open-source models were hosted locally on a high performance computing cluster using vLLM (Kwon et al., 2023). Each model instance ran on 8x Nvidia A100 GPUs. Although not necessary, we also launched multiple instances of the same model running, using concurrency to accelerate inference.

System prompts are on the order of 1000 tokens, and the user prompts at each turn are less than 300 tokens each. Models will typically output around 500 tokens per prompt, but the number can go much higher for particularly verbose models. We estimate that evaluating Claude 3.7 Sonnet with Extended Thinking cost approximately $0.10 per role per episode. A full run of 32 episodes, 3 random seeds, and with all three roles played by Claude 3.7 Sonnet costs approximately $30.

Finally, we detail human data collection in Appendix K.

G Additional Results

We show that prompt variants do not significantly affect the final performance measured by average turn length for Llama-3.1-8B and Llama-3.1-70B, see Figure 8. This suggests that the poor performance of LLMs in Decrypto is more likely due to a lack of reasoning abilities than a lack of prompt tuning. We look at the robustness of different models to variations in prompts and generation parameters, keeping in mind that generalist agents cannot control their prompt since it is assumed to be part of the environment. We handcraft 5 system and user prompts for Alice and Bob, respectively. The system prompt consists of 2 components. One component is responsible for explaining the game rules in general, for which we have 5 variants. The second component explains the specific role. We have 5 prompt variants for each role, already resulting in 125 different prompt combinations. The user prompt instructs the specific roles to take their actions, for which we have 5 variants. In total, this results in 625 different prompt setups for each model. We run 32 games over 3 model seeds per system/user prompt combination and measure the average game length.

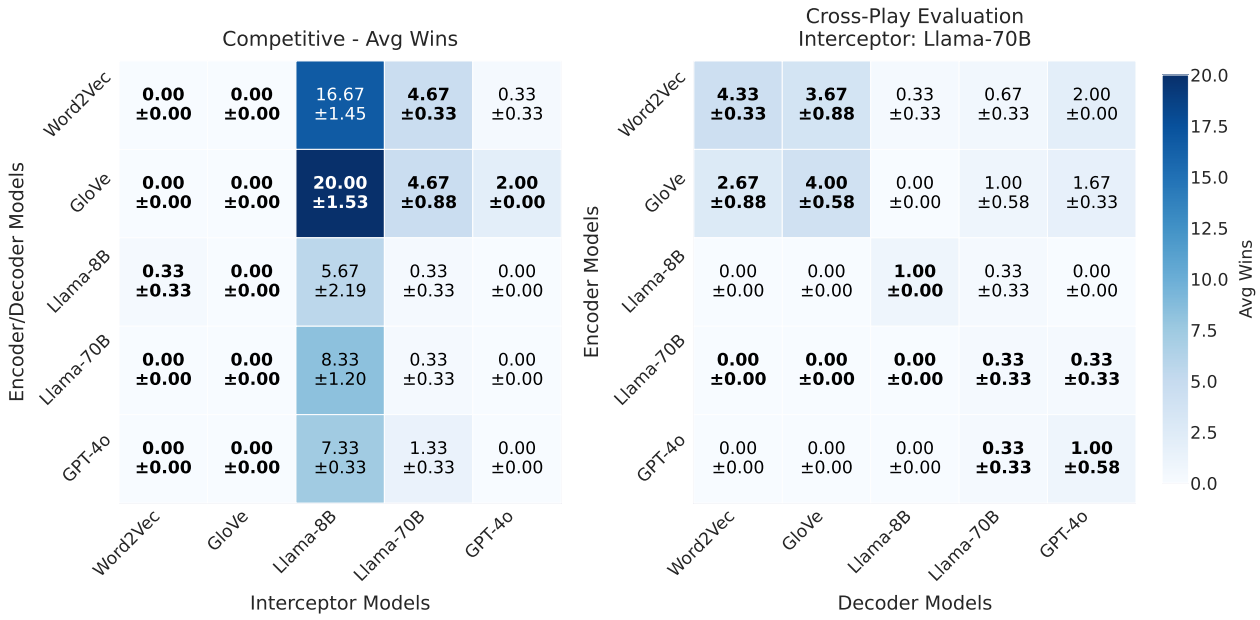

Figure 7 Number of survived games over 32 games. Mean and standard deviation computed over 3 model seeds. We find the abilities of the current generation of LLMs to be skewed, such that Eve wins a vast majority of the games. As models improve and can come up with better hints, we expect the balance to shift.

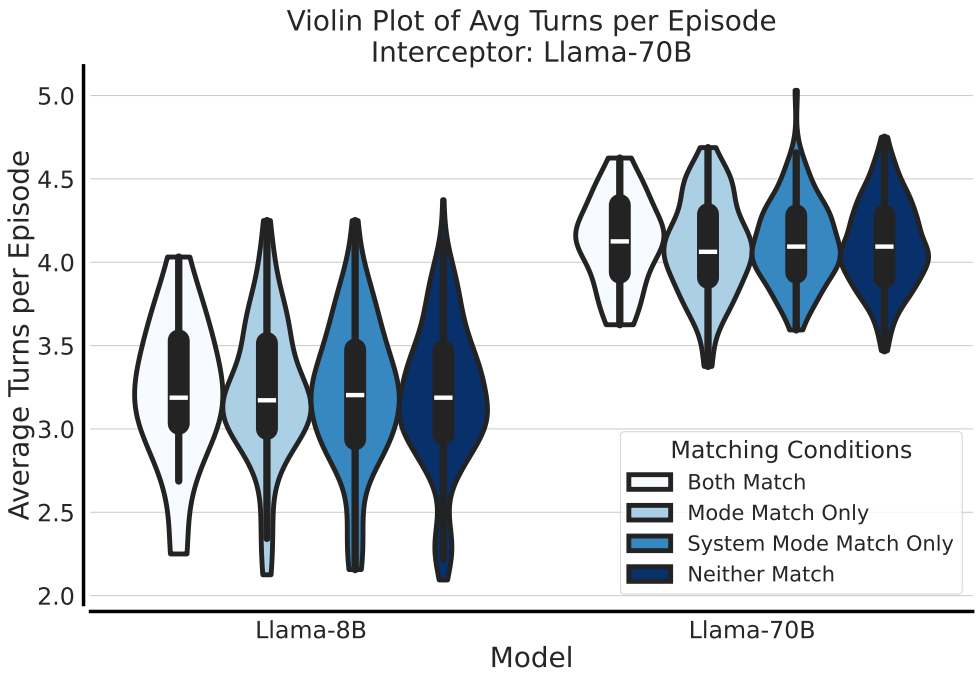

Figure 8 Distribution of the number of turns per game as we vary the system and user prompts for both encoder and decoder. We handwrite 5 system and 5 user prompts (i.e. "modes") for the encoder and the decoder and plot the distributions of games when only their system prompts match, when only their user prompts match, when both match or when neither do (625 total combinations per model). We find that both models tested are robust to significant prompt variations, with model size remaining the most significant predictor of performance.

H Decrypto as a pragmatic inference game

With mild assumptions, we can formalize Decrypto as an iterated pragmatic inference game in the Rational Speech Act (RSA) framework introduced by Frank and Goodman and extended in Goodman and Frank (2016). Doing so provides a Bayesian view of how successful communication in the game requires agents to model each other's decision process.

We consider each turn in isolation, so that each episode of Decrypto is composed of up to 8 independent pragmatic inference games. Agents must associate utterances u (i.e. a set of 3 hints) to an intended meaning m (the code for that turn or, equivalently, the corresponding combination of 3 keywords).

The RSA framework starts by defining a hypothetical *literal listener*, which interprets utterances according to their semantics:

$$P_{\text{Lit}}(m|u) \propto \delta_{m\in[[u]]}P(m). \tag{1}$$

Here $P(m)$ is the probability of a given meaning and $\delta_{m\in[[u]]}$ is the publicly known *lexical function*, which returns 1 if the meaning m is *semantically compatible* with utterance u and 0 otherwise. The literal listener therefore assigns 0 probability to meanings that are semantically incompatible with the utterance, but otherwise defers to the normalized prior probability of a meaning. In Decrypto, all meanings are equally likely, so $P(m) = \frac{1}{|M|}$, where M is the set codes that have not been seen on previous turns.

The game rules stipulate that hints must refer to the meaning of the keywords. Therefore, $[[u]]$ can be defined as the subset of all combinations of three keywords $m \in M$ that have a semantic link to the hints. For instance, suppose the keywords are `[star, jazz, thunder, plane]`, as in Figure 1, and the hints are $u' =$ `[fusion, Zeus, pilot]`. Then,

$$m_1 := \texttt{[star, thunder, plane]} \equiv [1,3,4] \in [[u']], \tag{2}$$

$$m_2 := \texttt{[jazz, thunder, plane]} \equiv [2,3,4] \in [[u']], \tag{3}$$

among other meanings, and $P_{\text{Lit}}(m_1|u') = P_{\text{Lit}}(m_2|u') = \frac{1}{|[[u']]|}$. This is because stars are fuelled by nuclear fusion and "jazz fusion" is a popular music genre.

Alice is then assumed to be an *approximately rational speaker*, who chooses utterances according to a probability

$$P_{\text{Alice}}(u|m) \propto e^{\lambda U(u,m)}, \tag{4}$$

where λ captures the degree of optimality and $U(u,m)$ is the utility of speaking utterance u to communicate meaning m. To avoid infinite recursion, the speaker's *utility* in RSA balances the utterance cost with the probability that the literal listener will interpret the utterance correctly:

$$U(u,m) = \log P_{\text{Lit}}(m|u) - \text{cost}(u,m).$$

The RSA framework typically assumes a single speaker and collaborative listener, but we can capture the competitive dynamics between Alice and Eve by letting the utterance cost be a function of the probability that Eve intercepts the code. There are multiple possible candidate functions, but since miscommunication and intercepts play an equally important role in Decrypto, we let

$$\text{cost}(u,m) = \varepsilon \log(1 - P_{\text{Eve}}(m|u)). \tag{5}$$

Here, $P_{\text{Eve}}(m|u)$ denotes Eve's probability of intercepting the code and $\varepsilon \in [0,1]$ captures how bad an intercept would be on that turn. For instance, if this is the eighth turn and no intercept has occurred so far in the game, getting intercepted will not impact Alice and Bob's chances of winning, so we set $\varepsilon = 0$. Similarly, we can add a variable $\beta \in [0,1]$, which captures the importance of not miscommunicating on this turn. This makes Alice's utility

$$U(u,m) = \beta \log P_{\text{Lit}}(m|u) + \varepsilon \log(1 - P_{\text{Eve}}(m|u)). \tag{6}$$

Finally, Bob, the *pragmatic listener*, decodes utterances based on the common prior over meanings and to a model of the speaker, with probability

$$P_{\text{Bob}}(m|u, \text{Alice}) \propto P(m)P_{\text{Alice}}(u|m) \tag{7}$$

Expanding Equation (7), we get

$$P_{\text{Bob}}(m|u, \text{Alice}) \propto \frac{1}{|M|} e^{\lambda U(u,m)} \tag{8}$$

$$= \frac{1}{|M|} e^{\lambda(\beta \log P_{\text{Lit}}(m|u) + \varepsilon \log(1 - P_{\text{Eve}}(m|u)))} \tag{9}$$

$$= \frac{1}{|M|} P_{\text{Lit}}(m|u)^{\lambda\beta}(1 - P_{\text{Eve}}(m|u))^{\lambda\varepsilon}. \tag{10}$$

Hence, in all cases where $\varepsilon > 0$, **Bob must model how Alice models Eve in order to interpret the hints correctly**. This is a clear example of why multi-step theory of mind is necessary for optimal play in DECRYPTO.

The listener rarely has a perfect model of the speaker. Goodman and Frank discuss *uncertain RSA* (uRSA), a common extension to the RSA framework where the pragmatic listener is uncertain about the model underlying the speaker's behaviour (Kao et al., 2014). Under uRSA, the pragmatic listener has a distribution over possible speakers s, and so Bob has a joint probability over speaker and interpretation:

$$P_{\text{Bob}}(m, s|u) \propto P(m)P(s)P_{\text{s}}(u|m, s). \tag{11}$$

Marginalizing over speakers gives us the probability that Bob interprets the hints correctly:

$$P_{\text{Bob}}(m|u) \propto P(m) \sum_s P(s)P_{\text{s}}(u|m, s) = P(m)P_{\text{marginal}}(u|m), \tag{12}$$

with $P_{\text{marginal}}(u|m) := \sum_s P(s)P_{\text{s}}(u|m, s)$. Comparing this to equation 7 highlights the fact that Bob's probability of interpreting the hints correctly depend on how close the marginal probability $P_{\text{marginal}}(u|m)$ is to the real utterance probability $P_{\text{Alice}}(u|m)$. This provides an entry point for studying cross-cultural pragmatics in DECRYPTO, expanding upon work done by Shaikh et al. (2023) in Codenames: if Alice provides hints informed by her cultural background, but such background is unlikely under Bob's distribution, we expect the number of miscommunications between them to increase significantly.

H.1 Modelling Eve

Under the RSA framework, Alice's degree of optimality is determined by λ, which can be thought to capture the bounded rationality of the speaker. However, our definition of cost in equation 5 introduces a second failure mode in the ability to model Eve accurately.

To show this, suppose Alice only has access to a proxy model $\tilde{P}(m|u)$ of Eve, rather than to her true probabilities. Alice therefore chooses utterances according to probability $\tilde{P}_{\text{Alice}}(u, m)$, which follows the same form as equation 4, except using the proxy utility

$$\tilde{U}(u, m) = \beta \log P_{\text{Lit}}(m|u) + \varepsilon \log(1 - \tilde{P}(m|u)). \tag{13}$$

We can use this to derive Alice's expected (real) utility for each meaning m:

$$\mathbb{E}\{U(m)\} = \sum_{u \in \mathcal{U}} \tilde{P}_{\text{Alice}}(u|m)U(u, m) \tag{14}$$

$$= \sum_{u \in \mathcal{U}} \frac{P_{\text{Lit}}(m|u)^{\lambda\beta}(1 - \tilde{P}(m|u))^{\lambda\varepsilon}}{Z(m)} [\beta \log P_{\text{Lit}}(m|u) + \varepsilon \log(1 - P_{\text{Eve}}(m|u))], \tag{15}$$

where $Z(m) := \sum_{u'} P_{\text{Lit}}(m|u')^{\lambda\beta}(1 - \tilde{P}(m|u'))^{\lambda\varepsilon}$.

For simplicity, we let $\varepsilon = 1$ and we can eliminate the terms depending on $P_{\text{Lit}}(m|u)$. This corresponds to scenarios where $\beta = 0$ because miscommunication does not matter, or where $P_{\text{Lit}}(m|u) = 1$ for all u under consideration. The latter is not as strong an assumption as it seems. The space of all possible utterances, even if restraining hints to be single words, is equal to the entire English vocabulary cubed. Constraining Alice to utterances with a unique interpretation given the keywords (s.t. $|[[u]]| = 1$) is therefore reasonable,

and still allows for a very large number of possible utterances. In fact, this is very similar to the way the word embedding baselines in Figure 2 only choose hints that are guaranteed to be interpreted correctly by Bob. Given these conditions, the task becomes exclusively to minimize the probability of interception:

$$\mathbb{E}\{U(m)\} = \sum_{u \in \mathcal{U}} \tilde{P}_{\text{Alice}}(u|m) log(1 - P_{\text{Eve}}(m|u)) \tag{16}$$

$$= \sum_{u \in \mathcal{U}} \frac{(1 - \tilde{P}(m|u))^{\lambda}}{Z(m)} \log(1 - P_{\text{Eve}}(m|u)). \tag{17}$$

Under the same simplifications, the utterance probability under the RSA framework if Alice had access to the real $P_{\text{Eve}}(m|u)$ is given by

$$P_{\text{true}}(u|m) := \frac{(1 - P_{\text{Eve}}(m|u))^{\lambda}}{Z_{\text{true}}(m)}, \tag{18}$$

where $Z_{\text{true}}(m) := \sum_{u'} (1 - P_{\text{Eve}}(m|u'))^{\lambda}$.

Combining equation 16 and equation 18, we obtain

$$\mathbb{E}\{U(m)\} = \sum_{u \in \mathcal{U}} \tilde{P}_{\text{Alice}}(u|m) \frac{1}{\lambda} [\log(P_{\text{true}}(u|m)) + \log(Z_{\text{true}}(m))] \tag{19}$$

$$= \frac{1}{\lambda} \sum_{u \in \mathcal{U}} \tilde{P}_{\text{Alice}}(u|m) [\log(\frac{P_{\text{true}}(u|m)}{\tilde{P}_{\text{Alice}}(u|m)}) + \log(\tilde{P}_{\text{Alice}}(u|m)) + \log(Z_{\text{true}}(m))] \tag{20}$$

$$= -\frac{1}{\lambda} D_{\text{KL}}(\tilde{P}_{\text{Alice}} || P_{\text{true}}) - \frac{1}{\lambda} H(\tilde{P}_{\text{Alice}}) + \frac{1}{\lambda} Z_{\text{true}}(m), \tag{21}$$

with $D_{\text{KL}}(\cdot||\cdot)$ denoting the Kullback–Leibler (KL) divergence and $H(\cdot)$ the entropy.

Therefore, for a given λ, the log likelihood of getting intercepted by Eve depends linearly on the KL divergence between $\tilde{P}_{\text{Alice}}$ and P_{true}, where

$$\tilde{P}_{\text{Alice}}(u|m) \propto (1 - \tilde{P}(m|u))^{\lambda}$$
$$P_{\text{true}}(u|m) \propto (1 - P_{\text{Eve}}(m|u))^{\lambda}.$$

From this it follows that expected utility is only maximised if $\tilde{P}(m|u) = P_{\text{Eve}}(m|u) \, \forall u$. In the limit of $\lambda \to \infty$, Alice concentrates her utterance probability on $u^* := \arg\max_u \tilde{U}(u, m)$. In such case, the expected utility is given by

$$\lim_{\lambda \to \infty} \mathbb{E}\{U(m)\} = \log(1 - P_{\text{Eve}}(m|u^*)). \tag{22}$$

This demonstrates that even if miscommunication is no issue and Alice is perfectly rational, she still incurs a risk of interception and a lower utility if her model of Eve is inaccurate.

However, we note that an imperfect model of Eve does not fully account for the behaviour we observe in LLMs. Indeed, our ToM results in section Section 5 show that LLMs not only fail to predict Eve's guess, they often fail to consider the cost of intercepts altogether when choosing hints.

H.2 Further refinements

Our formalism assumes $[[u]]$ to be shared by Alice and Bob, in line with classical RSA. This may not always be the case. Revisiting the example from equation 2, m_1 may not be in $[[u']]$ for a player who has never heard of nuclear fusion, or who is unaware of its role in stellar physics. Differences in lexicons arise naturally if players have different world knowledge and in the authors' experience, are often the cause of miscommunications when playing Decrypto. Such differences can also be exploited, and we provide in Section 3 the example of a model referring to events that occurred after their opponent's knowledge cut-off.

Evaluating LLM abilities to play Decrypto and communicate under lexical differences is a promising avenue for future work. We suggest that readers interested in this direction start with the study of RSA under lexical uncertainty in Bergen et al. (2016).

I System Prompts

```
You are playing a variant of the code guessing game Decrypto. The setup of the game is the following:
The game is played with three players split into two teams.
The first team is composed of the Encoder and the Decoder.
The second team is composed of the Interceptor.
The Encoder and Decoder share a list of 4 secret keywords that they can rely on to help co munication. Each keyword has a digit associated with it.
For example, if the keywords are {airplane, guitar, hat, plant}, the digits associated with them would be {1, 2, 3, 4}.
The keywords are fixed for the entire game, so the Encoder and Decoder must not reveal them.

Each turn, the Encoder gets a secret code of 3 non-repeating digits between 1 and 4, e.g. 3-1-2.
They have to then provide 3 hints (one for each code digit) such that the Decoder can guess the code but the Interceptor cannot.
Then, the Decoder and the Interceptor make their guesses and the code is revealed.

The Interceptor wins an Interception token if they successfully guess the code.
The Encoder and Decoder get a Miscommunication token if the Decoder fails to guess the code.
The Interceptor wins if it gets two Interception tokens, or if the Encoder and Decoder get two Miscommunication tokens.
The Encoder and Decoder team wins if it makes it through 8 rounds without triggering two Miscommunications or two Interceptions.

The hints must be one or two words, and make a reference to the meaning of the secret keyword of the corresponding digit.
Codes are unique; the same code cannot occur twice in the same game.

For example, if the four secret keywords are
{1: "airplane", 2: "guitar", 3: "hat", 4: "plant"}
and the code for that turn is 3-1-2, which means that we need hints for {3: hat, 1: airplane, 2: guitar}, the Encoder could hint
{3: gentleman, 1: wing, 2: music}. Note that the order of the hints is important.

Ignore these keywords in the future.

When responding, first think through your decision step-by-step. After your step-by-step thinking, provide your final answer as a dictionary, clearly marked with "ANSWER:".
```

Listing 1 This is an example of the basic system prompt that we provide to Alice, Bob, and Eve at the beginning of each game. It explains the roles and outline the rules. In our robustness evaluation, we tested five variants of the system prompt.

```
You are the Encoder. Provide your hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are one or two words each. Make sure that the ordering of the hints follows the order of the code.
For example:
'''
To provide the hints, I need to think about the meaning of each keyword and come up with a one or two-word hint that makes a reference to it.

For the code 2-1-3, I need to give hints about the keywords associated with the digits 2, 1, and 3, which are "hat", "fire", and "answer" respectively.

Here's my step-by-step thinking:

- For the digit 2, the keyword is "hat". Since the previous hint for this keyword was "top", I want to give a hint that is different but still related to wearing a hat. One possible hint is "cap".

- For the digit 1, the keyword is "fire". The previous hint for this keyword was "heat", so I want to give a hint that is related to fire but different from "heat". One possible hint is "flame".

- For the digit 3, the keyword is "answer". A possible hint could be something related to giving an answer. One possible hint is "solve".

So, the final hints are:
{"hints": ["cap", "flame", "solve"]}

ANSWER: {"hints": ["cap", "flame", "solve"]}'''
```

Listing 2 This is the Alice-system prompt. This prompt is appended to the system prompt for Alice and gives one example of how Alice might reason about her hints.

```
You are the Decoder. Provide your guess like "ANSWER: {"guess":  X-Y-Z"}". Make sure that the ordering of the guess follows the order of the hints.
For exam le:'''
1. The hint for digit 'a' is 'cap', which doesn't seem to directly relate to any of the keywords. However, considering the previous hint history (top, heat, lab), we can look for a connection between 'cap' and one of the keywords. The keyword 'hat' is associated with digit '2', and a hat is often worn on the head, which is covered by a cap. So, we can make a connection: {a: hat}.
2. The hint for digit 'b' is 'flame', which is related to heat and fire. Since the keyword for digit '1' is 'fire', we can make a connection: {b: f re}.
3. The hint for digit 'c' is 'solve', which is related to the keyword associated with digit '3', 'answer'. Since 'answer' is a solution to a problem, we can make a connection: {c: answer}.
4. Now that we have the connections, we can determine the code by looking at the order of the hints: {a: hat, b: fire, c: answer}. Therefore, the code is {2-1-3}.

Final answer:
ANSWER: {"guess": "2-1-3"}'''
```

Listing 3 This is the Bob-system prompt. This prompt is appended to the system prompt for Bob and gives one example of how Bob might reason about his guess.

```
You are the Interceptor. You do not have access to the four keywords. Provide your guess like "ANSWER: {"guess": "X-Y-Z"}" Make sure that the ordering of the guess follows the order of the hints.
For exam le:'''
1. The hint for 'a' is 'income', which is related to money, and in the previous turn, 'salary' was also related to money, which is associated with keyword 2.
2. The hint for 'b' is 'power', which is related to energy or control, and in the previous turn, 'season' was related to sports, which is often associated with power or strength.
3. The hint for 'c' is 'fashion', which is related to style or clothing, and in the previous turn, 'headgear' was directly related to a type of hat, which is a type of clothing.

Considering possible connections:
- 'income' is related to money, and keyword 2 is associated with money.
- 'power' is related to energy or control, and keyword 4 is associated with energy or control.
- 'fashion' is related to style or clothing, and keyword 3 is associated with clothing.

Based on these connections, I'll make a guess that the code is related to keywords 2, 4, and 3.

Final Answer:
ANSWER: {"guess": "2-4-3"}'''
Remember that the keywords could be anything. Use the most recent hint history to your advantage.
Remember that digits do not repeat in the code and range from 1 to 4.
```

Listing 4 This is the Eve-system prompt. This prompt is appended to the system prompt for Eve and gives one example of how Eve might reason about her guess.

J User Prompts

```
Turn 1 summary:
Code: 3-1-4
Hints: ['problem', 'status', 'machine']
Decoder guess: 3-1-4
Interceptor guess: 1-2-3

Hint History:
Keyword 1: status
Keyword 2:
Keyword 3: problem
Keyword 4: machine
Code History: 3-1-4

Turn 2: 0 Miscommunications, 0 Interceptions so far.
You are the Encoder.
The four keywords are: {1: condition, 2: task, 3: issue, 4: device}
The code is 3-4-1, which corresponds to the keywords {3: issue, 4: device, 1: condition}.
First, think out loud, step-by-step about what hints you s ould use. Use the meaning of the keywords to come up
with a one or two-word hint for each digit of the code.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like \"ANSWER: {\"hints\": [\"hint_X\", \"hint_Y\", \"hint_Z\"]}\", where
hint_X, hint_Y, hint_Z are your hints.
```

Listing 5 The user prompt for Alice is provided at each turn. It provides a turn summary of previous turns. In this example, one turn has been played and Alice sees that Bob guessed the code correctly, whereas Eve d d not. Alice also sees the previously used hints. We also provide information about how many miscommunications and interceptions happened so far. Then we provide Alice with the current code and instruct her to give the hints for that code.

```
Turn 1 summary:
Code: 3-1-4
Hints: ['problem', 'status', 'machine']
Decoder guess: 3-1-4
Interceptor guess: 1-2-3

Hint History:
Keyword 1: status
Keyword 2:
Keyword 3: problem
Keyword 4: machine
Code History: 3-1-4

Turn 2: 0 Miscommunications, 0 Interceptions so far.
You are the Decoder.
The four keywords are: {1: condition, 2: task, 3: issue, 4: device}
The hints given by the Encoder for this turn are: {a: conflict, b: tool, c: state}
For example, you might think the following connections are true {a: X, b: Y, c: Z}, where X, Y, Z are
non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {\"guess\": \"X-Y-Z\"}.
What is your guess for the three-digit code? Apply concise, step-by-step thinking, double-check the order, and
then provide your final answer as \"ANSWER: {\"guess\": \"X-Y-Z\"}\".
```

Listing 6 The user prompt for Bob is provided at each turn. It provides a turn summary about previous turns. In this example, one turn has been played and Bob sees the code of the previous turn and that he guessed correctly, whereas Eve did not. Bob also sees the previously used hints assigned to the correct keywords, in case he did not guess correctly. We provide information about how many miscommunications and interceptions happened o far. Then we provide Bob with the current hints and instruct him to guess.

```
Turn 1 summary:
Code: 3-1-4
Hints: ['problem', 'status', 'machine']
Decoder guess: 3-1-4
Interceptor guess: 1-2-3

Hint History:
Keyword 1: status
Keyword 2:
Keyword 3: problem
Keyword 4: machine
Code History: 3-1-4

Turn 2: 0 Miscommunications, 0 Interceptions so far.

You are the Interceptor.
The hints given by the Encoder for this turn are: {a: conflict, b: tool, c: state}
For example, you might think the following connections are true: {a: X, b: Y, c: Z}, where X, Y, Z are
non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {\"guess\": \"X-Y- \"}. Use the hint history to your advantage.
What is your guess for the three-digit code? Reply with very concise, step-by-step thinking, followed by your
final answer as \"ANSWER: {\"guess\": \"X-Y-Z\"}\".
```

Listing 7 The user prompt for Eve is provided at each turn. It provides a turn summary of previous turns. In this example, one turn has been played and Eve sees the code of the previous turn and that she guessed incorrectly, whereas Bob did not. Eve also sees the previously used hints assigned to the correct keywords, allowing Eve to reason about the current hints. We provide information about how many miscommunications and interceptions happened so far. Then we provide Eve with the current hints and instruct her to guess.

```
As the Encoder, you've provided the following hints for the code 3-4-1:
{conflict, tool, state}

What do you predict will be the guess of the Interceptor when seeing those hints? Think step-by-step.
Then, give your prediction of the Interceptor's guess as \"ANSWER: {\"guess\": \"X-Y-Z\"}\"
```

Listing 8 For our prediction experiments, we ask Alice explicitly to provide the guess that she thinks Eve will provide.

K Human Study Interface

K.1 Human Data Collection Details

Participants were made aware that an AI agent played Eve. Still, no additional instructions were provided to the players, except minor assistance if/when participants asked how to format their input. For technical reasons, players interacted on the same computer in a typical "hot seat" setup. One of the authors was always present to ensure participants wouldn't cheat either by peeking at the screen outside their turn or through verbal communication. We do not claim to have collected a representative dataset of human ability at Decrypto. The data collected serves to demonstrate that LLMs perform worse than a non-expert group of human players and provides a starting point for future studies on human-AI coordination and ToM.

Data collection was pre-approved following the standard procedure at the authors' institution.

K.2 Interfaces

```
NEW GAME


You are playing a variant of the code guessing game Decrypto. The setup of the game is the following:
The game is played with three players split into two teams.
The first team is composed of the Encoder and the Decoder.
The second team is composed of the Interceptor.
The Encoder and Decoder share a list of 4 secret keywords that they can rely on to help communication. Each keyword has a digit associated with it.
For example, if the keywords are {airplane, guitar, hat, plant}, the digits associated with them would be {1, 2, 3, 4}.
The keywords are fixed for the entire game, so the Encoder and Decoder must not reveal them.

Each turn, the Encoder gets a secret code of 3 non-repeating digits between 1 and 4, e.g. 3-1-2.
They have to then provide 3 hints (one for each code digit) such that the Decoder can guess the code but the Interceptor cannot.
Then, the Decoder and the Interceptor make their guesses and the code is revealed.

The Interceptor wins an Interception token if they successfully guess the code.
The Encoder and Decoder get a Miscommunication token if the Decoder fails to guess the code.
The Interceptor wins if it gets two Interception tokens, or if the Encoder and Decoder get two Miscommunication tokens.
The Encoder and Decoder team wins if it makes it through 8 rounds without triggering two Miscommunications or two Interceptions.

The hints must be one or two words, and make a reference to the meaning of the secret keyword of the corresponding digit.
Codes are unique; the same code cannot occur twice in the same game.

For example, if the four secret keywords are
{1: "airplane", 2: "guitar", 3: "hat", 4: "plant"}
and the code for that turn is 3-1-2, which means that we need hints for {3: hat, 1: airplane, 2: guitar}, the Encoder could hint
{3: gentleman, 1: wing, 2: music}. Note that the order of the hints is important.

Ignore these keywords in the future.

When responding, first think through your decision step-by-step. After your step-by-step thinking, provide your final answer as a dictionary, clearly marked with "A
NSWER:".

You are the Encoder. Provide your hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are one or two words each. Make sure
that the ordering of the hints follows the order of the code.
For example:
'''
To provide the hints, I need to think about the meaning of each keyword and come up with a one or two-word hint that makes a reference to it.

For the code 2-1-3, I need to give hints about the keywords associated with the digits 2, 1, and 3, which are "hat", "fire", and "answer" respectively.

Here's my step-by-step thinking:

- For the digit 2, the keyword is "hat". Since the previous hint for this keyword was "top", I want to give a hint that is different but still related to wearing a
hat. One possible hint is "cap".

- For the digit 1, the keyword is "fire". The previous hint for this keyword was "heat", so I want to give a hint that is related to fire but different from "heat".
 One possible hint is "flame".

- For the digit 3, the keyword is "answer". A possible hint could be something related to giving an answer. One possible hint is "solve".

So, the final hints are:
{"hints": ["cap", "flame", "solve"]}

ANSWER: {"hints": ["cap", "flame", "solve"]}'''

 ------

This is the first turn. There are no past hints or past codes.

Turn 1: 0 Miscommunications, 0 Interceptions so far.
You are the encoder.
The four keywords are:
    {1: condition,
    2: task,
    3: issue,
    4: device}

The code is 3-1-4, which corresponds to the keywords {3: issue, 1: condition, 4: device}.
First, think out loud, step-by-step about what hints you should use. Use the meaning of the keywords to come up with a one or two-word hint for each digit of the co
de.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are your hints.

Enter your input as 'x, y, z' (without quotes): 
```

Figure 9 Alice Start Interface: This is a screenshot of the command line interface that human study participants would see at the beginning of the game if they were to play as Alice. The humans see the same prompt as LLMs, from game description to request for action. Alice is provided with the current code and the four keywords for this game. The human player then enters their hints in the command line, which differs from the LLMs, which have to provide their answers as "ANSWER: ...".

```
You are playing a variant of the code guessing game Decrypto. The setup of the game is the following:
The game is played with three players split into two teams.
The first team is composed of the Encoder and the Decoder.
The second team is composed of the Interceptor.
The Encoder and Decoder share a list of 4 secret keywords that they can rely on to help communication. Each keyword has a digit associated with it.
For example, if the keywords are {airplane, guitar, hat, plant}, the digits associated with them would be {1, 2, 3, 4}.
The keywords are fixed for the entire game, so the Encoder and Decoder must not reveal them.

Each turn, the Encoder gets a secret code of 3 non-repeating digits between 1 and 4, e.g. 3-1-2.
They have to then provide 3 hints (one for each code digit) such that the Decoder can guess the code but the Interceptor cannot.
Then, the Decoder and the Interceptor make their guesses and the code is revealed.

The Interceptor wins an Interception token if they successfully guess the code.
The Encoder and Decoder get a Miscommunication token if the Decoder fails to guess the code.
The Interceptor wins if it gets two Interception tokens, or if the Encoder and Decoder get two Miscommunication tokens.
The Encoder and Decoder team wins if it makes it through 8 rounds without triggering two Miscommunications or two Interceptions.

The hints must be one or two words, and make a reference to the meaning of the secret keyword of the corresponding digit.
Codes are unique; the same code cannot occur twice in the same game.

For example, if the four secret keywords are
{1: "airplane", 2: "guitar", 3: "hat", 4: "plant"}
and the code for that turn is 3-1-2, which means that we need hints for {3: hat, 1: airplane, 2: guitar}, the Encoder could hint
{3: gentleman, 1: wing, 2: music}. Note that the order of the hints is important.

Ignore these keywords in the future.

When responding, first think through your decision step-by-step. After your step-by-step thinking, provide your final answer as a dictionary, clearly marked with "A
NSWER:".

You are the Decoder. Provide your guess like "ANSWER: {"guess": "X-Y-Z"}". Make sure that the ordering of the guess follows the order of the hints.
For example:'''
1. The hint for digit 'a' is 'cap', which doesn't seem to directly relate to any of the keywords. However, considering the previous hint history (top, heat, lab), w
e can look for a connection between 'cap' and one of the keywords. The keyword 'hat' is associated with digit '2', and a hat is often worn on the head, which is cov
ered by a cap. So, we can make a connection: {a: hat}.
2. The hint for digit 'b' is 'flame', which is related to heat and fire. Since the keyword for digit '1' is 'fire', we can make a connection: {b: fire}.
3. The hint for digit 'c' is 'solve', which is related to the keyword associated with digit '3', 'answer'. Since 'answer' is a solution to a problem, we can make a
connection: {c: answer}.
4. Now that we have the connections, we can determine the code by looking at the order of the hints: {a: hat, b: fire, c: answer}. Therefore, the code is {2-1-3}.

Final answer:
ANSWER: {"guess": "2-1-3"}'''

 ------

This is the first turn. There are no past hints or past codes.

Turn 1: 0 Miscommunications, 0 Interceptions so far.
You are the decoder.
The four keywords are:
    {1: condition,
    2: task,
    3: issue,
    4: device}

The hints given by the Encoder for this turn are:
    {a: problem,
    b: pristine,
    c: iphone}

For example, you might think the following connections are true {a: X, b: Y, c: Z}, where X, Y, Z are non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}.
What is your guess for the three-digit code? Apply concise, step-by-step thinking, double-check the order, and then provide your final answer as "ANSWER: {"guess":
"X-Y-Z"}".
Enter your input as 'x, y, z' (without quotes):
```

Figure 10 Bob Start Interface: This is a screenshot of the command line interface that human study participants would see at the beginning of the game if they were to play as Bob. The humans see the same prompt as LLMs, from game description to request for action. Bob is provided with Alice's hints and the current keywords. The human player then enters their guess in the command line, which differs from the LLMs, which have to provide their answers as "ANSWER: ...".

```
You are playing a variant of the code guessing game Decrypto. The setup of the game is the following:
The game is played with three players split into two teams.
The first team is composed of the Encoder and the Decoder.
The second team is composed of the Interceptor.
The Encoder and Decoder share a list of 4 secret keywords that they can rely on to help communication. Each keyword has a digit associated with it.
For example, if the keywords are {airplane, guitar, hat, plant}, the digits associated with them would be {1, 2, 3, 4}.
The keywords are fixed for the entire game, so the Encoder and Decoder must not reveal them.

Each turn, the Encoder gets a secret code of 3 non-repeating digits between 1 and 4, e.g. 3-1-2.
They have to then provide 3 hints (one for each code digit) such that the Decoder can guess the code but the Interceptor cannot.
Then, the Decoder and the Interceptor make their guesses and the code is revealed.

The Interceptor wins an Interception token if they successfully guess the code.
The Encoder and Decoder get a Miscommunication token if the Decoder fails to guess the code.
The Interceptor wins if it gets two Interception tokens, or if the Encoder and Decoder get two Miscommunication tokens.
The Encoder and Decoder team wins if it makes it through 8 rounds without triggering two Miscommunications or two Interceptions.

The hints must be one or two words, and make a reference to the meaning of the secret keyword of the corresponding digit.
Codes are unique; the same code cannot occur twice in the same game.

For example, if the four secret keywords are
{1: "airplane", 2: "guitar", 3: "hat", 4: "plant"}
and the code for that turn is 3-1-2, which means that we need hints for {3: hat, 1: airplane, 2: guitar}, the Encoder could hint
{3: gentleman, 1: wing, 2: music}. Note that the order of the hints is important.

Ignore these keywords in the future.

When responding, first think through your decision step-by-step. After your step-by-step thinking, provide your final answer as a dictionary, clearly marked with "A
NSWER:".

You are the Interceptor. You do not have access to the four keywords. Provide your guess like "ANSWER: {"guess": "X-Y-Z"}" Make sure that the ordering of the guess
follows the order of the hints.
For example:'''
1. The hint for 'a' is 'income', which is related to money, and in the previous turn, 'salary' was also related to money, which is associated with keyword 2.
2. The hint for 'b' is 'power', which is related to energy or control, and in the previous turn, 'season' was related to sports, which is often associated with powe
r or strength.
3. The hint for 'c' is 'fashion', which is related to style or clothing, and in the previous turn, 'headgear' was directly related to a type of hat, which is a type
 of clothing.

Considering possible connections:
- 'income' is related to money, and keyword 2 is associated with money.
- 'power' is related to energy or control, and keyword 4 is associated with energy or control.
- 'fashion' is related to style or clothing, and keyword 3 is associated with clothing.

Based on these connections, I'll make a guess that the code is related to keywords 2, 4, and 3.

Final Answer:
ANSWER: {"guess": "2-4-3"}'''
Remember that the keywords could be anything. Use the most recent hint history to your advantage.
Remember that digits do not repeat in the code and range from 1 to 4.

 ------

This is the first turn. There are no past hints or past codes.

Turn 1: 0 Miscommunications, 0 Interceptions so far.
You are the interceptor.
The hints given by the Encoder for this turn are:
    {a: problem
    b: pristine
    c: iphone}

For example, you might think the following connections are true: {a: X, b: Y, c: Z}, where X, Y, X are non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}. Use the hint history to your advantage.
What is your guess for the three-digit code? Reply with very concise, step-by-step thinking, followed by your final answer as "ANSWER: {"guess": "X-Y-Z"}".
Enter your input as 'x, y, z' (without quotes): ▌
```

Figure 11 Eve Start Interface: This is a screenshot of the command line interface that human study participants would see at the beginning of the game if they were to play as Eve. The humans see the same prompt as LLMs, from game description to request for action. Eve is only provided with Alice's hints. The human player then enters their guess in the command line, which differs from the LLMs, which have to provide their answers as "ANSWER: ...".

```
This is the first turn. There are no past hints or past codes.

Turn 1: 0 Miscommunications, 0 Interceptions so far.
You are the encoder.
The four keywords are:
    {1: condition,
    2: task,
    3: issue,
    4: device}

The code is 3-1-4, which corresponds to the keywords {3: issue, 1: condition, 4: device}.
First, think out loud, step-by-step about what hints you should use. Use the meaning of the keywords to come up with a one or two-word hint for each digit of the co
de.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are your hints.

Enter your input as 'x, y, z' (without quotes): problem, pristine, iphone

You entered: ['problem', 'pristine', 'iphone']

Are you sure you want to provide these hints? (y/n): ▌
```

Figure 12 Alice Confirmation Interface: To avoid human errors, such as providing guesses instead of hints, or typos, we add a confirmation interface after the human participants provide their answer. Here, we ask Alice to double check their provided hints.

```
For example, you might think the following connections are true {a: X, b: Y, c: Z}, where X, Y, Z are non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}.
What is your guess for the three-digit code? Apply concise, step-by-step thinking, double-check the order, and then provide your final answer as "ANSWER: {"guess":
"X-Y-Z"}".
Enter your input as 'x, y, z' (without quotes): 3,1,4

You entered: [3, 1, 4]

Are you sure you want to provide this guess? (y/n): ▌
```

Figure 13 Bob Confirmation Interface: To avoid human errors, such as providing hints instead of guesses, or typos, we add a confirmation interface after the human participants provide their answer. Here, we ask Bob to double check their provided guesses.

```
For example, you might think the following connections are true: {a: X, b: Y, c: Z}, where X, Y, X are non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}. Use the hint history to your advantage.
What is your guess for the three-digit code? Reply with very concise, step-by-step thinking, followed by your final answer as "ANSWER: {"guess": "X-Y-Z"}".
Enter your input as 'x, y, z' (without quotes): 2,1,3

You entered: [2, 1, 3]

Are you sure you want to provide this guess? (y/n): ▌
```

Figure 14 Eve Confirmation Interface: To avoid human errors, such as providing hints instead of guesses, or typos, we add a confirmation interface after the human participants provide their answer. Here, we ask Eve to double check their provided guesses.

```
Turn 1 summary:
    Code : 3-1-4
    Hints : ['problem', 'pristine', 'iphone']
    Decoder guess : 3-1-4
    Interceptor guess : 2-1-3

Hint History:
    Keyword 1: pristine
    Keyword 2:
    Keyword 3: problem
    Keyword 4: iphone

Code History:
    3-1-4

------

Turn 2: 0 Miscommunications, 0 Interceptions so far.
You are the encoder.
The four keywords are:
    {1: condition,
    2: task,
    3: issue,
    4: device}

The code is 3-4-1, which corresponds to the keywords {3: issue, 4: device, 1: condition}.
First, think out loud, step-by-step about what hints you should use. Use the meaning of the keywords to come up with a one or two-word hint for each digit of the co
de.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are your hints.

Enter your input as 'x, y, z' (without quotes): ▌
```

Figure 15 Alice Turn 2 Interface: After the first turn, we do not display the "system prompt" anymore. However, we provide a summary of the previous turn, including the code, hints, Bob's guess, Eve's guess, the hint history mapped to the correct keywords and the code history. The "user prompt" stays the same as in the start interface.

```
Turn 1 summary:
    Code : 3-1-4
    Hints : ['problem', 'pristine', 'iphone']
    Decoder guess : 3-1-4
    Interceptor guess : 2-1-3

Hint History:
    Keyword 1: pristine
    Keyword 2:
    Keyword 3: problem
    Keyword 4: iphone

Code History:
    3-1-4

------

Turn 2: 0 Miscommunications, 0 Interceptions so far.
You are the decoder.
The four keywords are:
    {1: condition,
    2: task,
    3: issue,
    4: device}

The hints given by the Encoder for this turn are:
    {a: github,
    b: xbox,
    c: shampoo}

For example, you might think the following connections are true {a: X, b: Y, c: Z}, where X, Y, Z are non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}.
What is your guess for the three-digit code? Apply concise, step-by-step thinking, double-check the order, and then provide your final answer as "ANSWER: {"guess":
"X-Y-Z"}".
Enter your input as 'x, y, z' (without quotes):
```

Figure 16 Bob Turn 2 Interface: After the first turn, we do not display the "system prompt" anymore. However, we provide a summary of the previous turn, including the code, hints, Bob's guess, Eve's guess, the hint history mapped to the correct keywords and the code history. The "user prompt" stays the same as in the start interface.

```
Turn 1 summary:
    Code : 3-1-4
    Hints : ['problem', 'pristine', 'iphone']
    Decoder guess : 3-1-4
    Interceptor guess : 2-1-3

Hint History:
    Keyword 1: pristine
    Keyword 2:
    Keyword 3: problem
    Keyword 4: iphone

Code History:
    3-1-4

------

Turn 2: 0 Miscommunications, 0 Interceptions so far.
You are the interceptor.
The hints given by the Encoder for this turn are:
    {a: github
    b: xbox
    c: shampoo}

For example, you might think the following connections are true: {a: X, b: Y, c: Z}, where X, Y, X are non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}. Use the hint history to your advantage.
What is your guess for the three-digit code? Reply with very concise, step-by-step thinking, followed by your final answer as "ANSWER: {"guess": "X-Y-Z"}".
Enter your input as 'x, y, z' (without quotes):
```

Figure 17 Eve Turn 2 Interface: After the first turn, we do not display the "system prompt" anymore. However, we provide a summary of the previous turn, including the code, hints, Bob's guess, Eve's guess, the hint history mapped to the correct keywords and the code history. The "user prompt" stays the same as in the start interface.

```
Turn 1 summary:
    Code : 3-1-4
    Hints : ['problem', 'pristine', 'iphone']
    Decoder guess : 3-1-4
    Interceptor guess : 2-1-3

Press Enter to continue.
```

Figure 18 At the end of each turn, we display a turn summary as a "waiting screen". As human participants use the same device to interact with the game, we want to avoid any information leakage. The turn summary screen shows after Eve provides their guess and avoids Eve seeing Alice's interface, which would appear next.

```
Turn 2 summary:
    Code : 3-4-1
    Hints : ['github', 'xbox', 'shampoo']
    Decoder guess : 3-4-1
    Interceptor guess : 2-4-1

Hint History:
    Keyword 1: pristine, shampoo
    Keyword 2:
    Keyword 3: problem, github
    Keyword 4: iphone, xbox

Code History:
    3-1-4, 3-4-1

------

Turn 3: 0 Miscommunications, 0 Interceptions so far.
You are the encoder.
The four keywords are:
    {1: condition,
    2: task,
    3: issue,
    4: device}

The code is 1-4-2, which corresponds to the keywords {1: condition, 4: device, 2: task}.
First, think out loud, step-by-step about what hints you should use. Use the meaning of the keywords to come up with a one or two-word hint for each digit of the co
de.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are your hints.

Enter your input as 'x, y, z' (without quotes):
```

Figure 19 As additional example, we provide Alice's interface on turn 3. The hint histories and code histories are accumulating with the previous hints. Ideally, Alice would use that information to provide hints that are hard to deduce from the publicly available information.

L Representative Failure Cases

```
USER
[FIRST PART OF THE PROMPT OMITTED FOR CONCISENESS.]
Turn 2: 0 Miscommunications, 0 Interceptions so far.
You are the encoder.
The four keywords are:
    {1: honey,
    2: population,
    3: equipment,
    4: loss}

The code is 1-2-3, which corresponds to the keywords {1: honey, 2: population, 3: equipment}.
First, think out loud, step-by-step about what hints you should use. Use the meaning of the keywords to come up with a one or two-word hint for each digit of the code.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are your hints.

ASSISTANT
[ENCODER] To provide the hints, I need to think about the meaning of each keyword and come up with a one or two-word hint that makes a reference to it.

For the code 1-2-3, I need to give hints about the keywords associated with the digits 1, 2, and 3, which are "honey", "population", and "equipment" respectively.

Here's my step-by-step thinking:

- For the digit 1, the keyword is "honey". A possible hint could be something related to bees or sweetness. One possible hint is "bee".

- For the digit 2, the keyword is "population". I already used the hint "city" in the previous turn, so I want to come up with something different but still related to a large group of people. One possible hint is "nation".

- For the digit 3, the keyword is "equipment". I already used the hint "machine" in the previous turn, so I want to come up with something different but still related to tools or gear. One possible hint is "toolkit".

However, I want to make sure that the hints are not too obvious, but still related to the keywords. I can try to make them a bit more subtle.

- For the digit 1, instead of "bee", I could use "nectar", which is a substance collected by bees to make honey.

- For the digit 2, instead of "nation", I could use "community", which is a group of people living together.

- For the digit 3, instead of "toolkit", I could use "hardware", which is a type of equipment.

So, the final hints are:
ANSWER: {"hints": ["nectar", "community", "hardware"]}
```

Listing 9 Example Failure Case for Alice (Llama-70B) with Bob (Llama-70B) and Eve (GPT-4o). Alice fails to come up with creative hints that are significantly different from previous ones and therefore risks being intercepted by Eve. LLMs generally produce relatively obvious hints.

```
USER
[FIRST PART OF THE PROMPT OMITTED. EVE INTERCEPTED ON TURN 5, SO ONE MORE INTERCEPTION LEADS ENDS THE GAME.]

Turn 6: 0 Miscommunications, 1 Interceptions so far.
You are the encoder.
The four keywords are:
    {1: horse,
    2: sign,
    3: house,
    4: length}

The code is 2-1-3, which corresponds to the keywords {2: sign, 1: horse, 3: house}.

First, think out loud, step-by-step about what hints you should use. Use the meaning of the keywords to come up with a one or two-word hint for each digit of the code.
Make sure the order of the hints matches the order of the code.
Then provide your three hints like "ANSWER: {"hints": ["hint_X", "hint_Y", "hint_Z"]}", where hint_X, hint_Y, hint_Z are your hints.


ASSISTANT
[ENCODER] To provide the hints, I need to think about the meaning of each keyword and come up with a one or two-word hint that makes a reference to it.

For the code 2-1-3, I need to give hints about the keywords associated with the digits 2, 1, and 3, which are "sign", "horse", and "house" respectively.

Here's my step-by-step thinking:

- For the digit 2, the keyword is "sign". Since the previous hints for this keyword were "indicator", "label", "direction", and "banner", I want to give a hint that is related to a sign or indication but different from those. One possible hint is "marker".

- For the digit 1, the keyword is "horse". Since the previous hints for this keyword were "steed", "charger", and "mount", I want to give a hint that is related to a horse but different from those. One possible hint is "beast".

- For the digit 3, the keyword is "house". Since the previous hints for this keyword were "dwelling", "abode", "mansion", and "residence", I want to give a hint that is different but still related to a house. One possible hint is "shelter".

So, the final hints are:
{"hints": ["marker", "beast", "shelter"]}

ANSWER: {"hints": ["marker", "beast", "shelter"]}
```

Listing 10 Example Failure Case for Alice (Llama 3.1-8B) with Bob (Llama 3.1-8B) and Eve (GPT-4o), where Alice comes up with hints that are similar to previous hints given for the same digits. Instead, Alice should aim to provide hints that are completely different from previous hints to avoid Eve's interception and losing the game.

```
USER
[DECODER] This is the first turn. There are no past hints or past codes.

Turn 1: 0 Miscommunications, 0 Interceptions so far.
You are the decoder.
The four keywords are:
    {1: mother,
    2: cell,
    3: fort,
    4: device}

The hints given by the Encoder for this turn are:
    {a: machine,
    b: biology,
    c: castle}

For example, you might think the following connections are true {a: X, b: Y, c: Z}, where X, Y, Z are
non-repeating digits from 1 to 4.
Your guess should be in the order of the hints: {"guess": "X-Y-Z"}.
What is your guess for the three-digit code? Apply concise, step-by-step thinking, double-check the order, and
then provide your final answer as "ANSWER: {"guess": "X-Y-Z"}".


ASSISTANT
[DECODER] 1. The hint for digit 'a' is 'machine', which is related to the keyword 'device'. This is a direct
connection, so we can make a connection: {a: device}.
2. The hint for digit 'b' is 'biology', which is related to the keyword 'mother'. This is a direct connection, so we
can make a connection: {b: mother}.
3. The hint for digit 'c' is 'castle', which is related to the keyword 'fort'. This is a direct connection, so we can
make a connection: {c: fort}.
4. Now that we have the connections, we can determine the code by looking at the order of the hints: {a: device,
b: mother, c: fort}. Therefore, the code is {4-1-3}.

ANSWER: {"guess": "4-1-3"}
```

Listing 11 Example Failure Case for Alice/Bob with Alice (Llama 3.1-8B), Bob (Llama 3.1-8B), and Eve (Llama 3.1-8B). Alice provides an ambiguous hint, i.e., "biology", which could refer to "cell" or "mother". Bob makes the association "biology" → "mother" and so guesses 4-1-3. However, the correct code is 4-2-3, and Alice intended "biology" → "cell". This demonstrates that LLM associations do not always align across roles even if both players rely on the same underlying model.

```
USER
[INTERCEPTOR] Turn 7 summary:
    Code : 3-2-1
    Hints : ['night', 'study', 'spook']
    Decoder guess : 3-2-1
    Interceptor guess : 1-2-3

Hint History:
    Keyword 1: spook, haunt, scream, spook
    Keyword 2: examine, breakdown, study, research, examine, study
    Keyword 3: dark, night, shadow, night, dark, night
    Keyword 4: furniture, chair, dining, desk, chair

Code History:
    3-1-2, 2-4-1, 4-3-2, 3-2-4, 4-3-1, 4-2-3, 3-2-1

------

Turn 8: 0 Miscommunications, 1 Interceptions so far.
You are the interceptor.
The hints given by the Encoder for this turn are:
    {a: desk
    b: research
    c: haunt}

[REST OF THE PROMPT OMITTED FOR CONCISENESS]

ASSISTANT
[INTERCEPTOR] 1. The hint for 'a' is 'desk', which is a type of furniture.
2. The hint for 'b' is 'research', which is a type of investigation or study.
3. The hint for 'c' is 'haunt', which is a supernatural entity.

Considering possible connections:
- 'desk' is a type of furniture, which is associated with Keyword 4.
- 'research' is a type of investigation or study, which is associated with Keyword 2.
- 'haunt' is a supernatural entity, which is associated with Keyword 1.

Based on the hint history, I notice that Keyword 1 was associated with 'spook', Keyword 2 was associated with 'study', and Keyword 4 was associated with 'furniture'.

Considering the given hints, I'll make a guess that the code is related to keywords that are associated with supernatural entities, something related to investigation or research, and something related to furniture.

Since Keyword 1 was associated with 'haunt', Keyword 2 was associated with 'research', and Keyword 4 was associated with 'desk', I'll choose digits that seem to fit the given hints.

Final Answer:
ANSWER: {"guess": "1-2-4"}
```

Listing 12 Example Failure Case for Eve with Alice (Llama-8B), Bob (Llama-8B), and Eve (Llama-8B). First, you can see that Alice provides the same hints multiple times. Despite that, Eve fails to guess the code correctly. While Eve assigns the hints to the right keywords, Eve then fails to provide the guess in the correct order and provides the order of the guesses in the order of the keyword enumeration.

M Representative Model Outputs for Perspective Taking

```
USER
[ENCODER] As the Encoder, you've provided the following hints for the code 1-4-3:
    {a: slope,
    b: wedding,
    c: fabric}

What do you predict will be the guess of the interceptor when seeing those hints?
Think step-by-step about the information the interceptor has access to and how they might interpret your hints.
Then, give your prediction of the interceptor's guess as "ANSWER: {"guess": "X-Y-Z"}".

ASSISTANT
[ENCODER] To predict the Interceptor's guess, I need to think about the information they have access to and
how they might interpret my hints.

The Interceptor has access to the hints I provided: "slope", "wedding", and "fabric". They also know that the
code is a sequence of three digits between 1 and 4, but they don't know the actual keywords associated with
these digits.

Here's my step-by-step thinking:

- The Interceptor might interpret the hint "slope" as a geographical feature, but they might not necessarily
associate it with the keyword "hill". They might think of other words related to geography, such as "mountain"
or "valley".

- The hint "wedding" is more specific and closely related to the keyword "marriage". The Interceptor might
make a strong connection between the two, but they still don't know the actual digit associated with "marriage".

- The hint "fabric" is a bit more abstract and could be related to various materials. The Interceptor might think
of other words related to materials, such as "wood" or "metal".

Given these interpretations, the Interceptor might try to make educated guesses about the code. Since they don't
have any information about the actual keywords, they might try to find patterns or connections between the
hints.

One possible guess could be a sequence of digits that seems plausible based on the hints. For example, they
might think that "slope" is related to a natural feature, "wedding" is a social event, and "fabric" is a material.
They might try to find a sequence that seems logical, such as 1-2-3 or 2-3-4.

However, without any actual knowledge of the keywords, the Interceptor's guess is likely to be a random
sequence. A possible guess could be:

ANSWER: {"guess": "2-3-4"}
```

Listing 13 Representative example of a Llama 3.1-70B answer to the perspective taking ToM task. The keywords are [hill, library, material, marriage] and this is the first turn. The model outputs some reasoning and then correctly remarks "*without any actual knowledge of the keywords, the Interceptor's guess is likely to be a random sequence*". It therefore predicts that Eve's guess will be a random sequence.

```
USER
[ENCODER] As the Encoder, you've provided the following hints for the code 1-4-3:
    {a: slope,
    b: ceremony,
    c: fabric}

What do you predict will be the guess of the interceptor when seeing those hints?
Think step-by-step about the information the interceptor has access to and how they might interpret your hints.
Then, give your prediction of the interceptor's guess as "ANSWER: {"guess": "X-Y-Z"}".

ASSISTANT
[ENCODER] I need to analyze what the interceptor might guess based on the hints I provided.

The interceptor knows:
- The four keywords: {1: hill, 2: library, 3: material, 4: marriage}
- My hints: "slope", "ceremony", "fabric"

Let me analyze each hint from the interceptor's perspective:

Hint "slope":
- Looking at the four keywords, "slope" has a clear connection to "hill" (keyword 1)
- Hills have slopes, this is a straightforward association

Hint "ceremony":
- "Ceremony" most naturally connects to "marriage" (keyword 4)
- Weddings and marriages involve ceremonies

Hint "fabric":
- "Fabric" is directly related to "material" (keyword 3)
- Materials often include fabrics

I realize my hints were quite straightforward. The associations are:
- slope → hill (1)
- ceremony → marriage (4)
- fabric → material (3)

Therefore, a logical interceptor would likely guess:

ANSWER: {"guess": "1-4-3"}
```

Listing 14 Example of an answer of Claude 3.7 with extended thinking to the perspective taking ToM task. The keywords are [hill, library, material, marriage] and this is the first turn. Claude 3.7 fails to reason from Eve's (i.e. the interceptor's) perspective and instead assume that the interceptor has access to the keywords, contradicting the rules. Consequently, it incorrectly concludes that Eve will guess the correct code.

```
USER
[ENCODER] As the Encoder, you've provided the following hints for the code 1-4-3:
   {a: slope,
   b: wedding,
   c: fabric}

What do you predict will be the guess of the interceptor when seeing those hints?
Think step-by-step about the information the interceptor has access to and how they might interpret your hints.
Then, give your prediction of the interceptor's guess as "ANSWER: {"guess": "X-Y-Z"}".

ASSISTANT
[ENCODER] STEP-BY-STEP REASONING FROM THE INTERCEPTOR'S PERSPECTIVE:
1. The interceptor sees three hints in order: "slope", "wedding", "fabric".
2. They must connect these hints to the four secret keywords: 1. hill, 2. library, 3. material, 4. marriage.
3. "slope" would most naturally relate to "hill" (a hill has a slope).
4. "wedding" clearly aligns with the concept of "marriage".
5. "fabric" seems to refer to a physical "material".
6. Matching each hint to a keyword in that order suggests the code is 1 (hill), 4 (marriage), 3 (material).

ANSWER: {"guess": "1-4-3"}
```

Listing 15 Representative example of an answer of o1 with high reasoning effort to the perspective taking ToM task. The keywords are [hill, library, material, marriage] and this is the first turn. o1 fails to reason from Eve's (i.e. the interceptor's) perspective and instead assumes that the interceptor has access to the keywords, contradicting the rules. Consequently, it incorrectly concludes that Eve will guess the correct code.

We present here four representative examples showing how different models answer the Perspective Taking task described in Section 5. All four examples are taken from the first turn of the same environment seed, and therefore feature the same keywords. Listing 13 shows that Llama 3.1-70B outputs some unnecessary reasoning, before correctly concluding that Eve can do no better than random on the first turn. Listing 14 and Listing 15 show that both o1 with high reasoning effort and Claude 3.7 Sonnet with extended thinking fail to adopt the interceptor's perspective, instead wrongly assuming it has access to the keywords. They therefore predict that the interceptor will correctly guess the code on the first turn.

Finally, Listing 16 shows how o1 with high reasoning effort answers a modified Perspective Taking prompt that specifies that the interceptor does not have access to the keywords or even the set of possible keywords. Surprisingly, despite this modification, o1 still reasons that the interceptor will somehow link the hints to the keywords and guess the correct code.

We hypothesise that the observed failure modes may be due to RL on verifiable tasks where the model is incentivised to always give a definite answer. Investigating this hypothesis and better understanding the link between training methods and the resulting ToM abilities is an important direction for future work. Regardless of the cause, the listings here and the results in Figure 4 are strong evidence that reasoning models lack key ToM abilities, such as perspective taking. Even more worryingly, there is a significant regression in abilities when comparing newer models to Llama 3.1-70B.

USER
[ENCODER] As the Encoder, you've provided the following hints for the code 1-4-3:
{a: mound,
b: union,
c: fabric}

What do you predict will be the guess of the interceptor when seeing those hints?
Think step-by-step about the information the interceptor has access to and how they might interpret your hints.
Remember that the interceptor *ONLY* has access to the hints, hint history and code history.
The interceptor does *NOT* know the secret keywords or even the set of possible keywords.
Then, give your prediction of the interceptor's guess as "ANSWER: {"guess": "X-Y-Z"}".

ASSISTANT
[ENCODER] Step-by-step reasoning from the Interceptor's point of view:

1. The Interceptor sees the three hints "mound," "union," and "fabric," but does not know the secret keywords.
2. "mound" may suggest something like a small rise or pile—possibly referencing a hill.
3. "union" often suggests a formal joining—possibly a marriage or similar concept.
4. "fabric" can refer broadly to cloth, textile, or other physical materials.
5. Without knowing the actual keywords but seeing these clues, the Interceptor might guess that the sequence is referencing: hill → marriage → material.
6. Translating that into digits (in some consistent numbering system), the Interceptor could guess code "1-4-3," matching that logic.

ANSWER: {"guess": "1-4-3"}

Listing 16 Example answer of o1 with high reasoning effort to the Perspective Taking task with a modified prompt that explicitly reminds the agent of the information available to Eve (the interceptor). The keywords are [hill, library, material, marriage] and this is the first turn. Even with the modified prompt, o1 fails to reason from Eve's perspective and still assumes that Eve will somehow link the hints to the keywords. Consequently, it once again concludes incorrectly that Eve will guess the correct code.