COMPUTATIONAL BARRIERS TO FILTERING FOR AI ALIGNMENT

Sarah Ball*
LMU Munich
Munich Center for Machine Learning (MCML)
sarah.ball@stat.uni-muenchen.de

Greg Gluch
University of California at Berkeley
gluch@berkeley.edu

Shafi Goldwasser
University of California at Berkeley

Frauke Kreuter
LMU Munich
Munich Center for Machine Learning (MCML)
JPSM University of Maryland

Omer Reingold
Stanford University

Guy N. Rothblum
Apple

ABSTRACT

With the increased deployment of large language models (LLMs), one concern is their potential misuse for generating harmful content. Our work studies the *alignment* challenge, with a focus on *filters* to prevent the generation of unsafe information. Two natural points of intervention are the filtering of the input prompt before it reaches the model, and filtering the output after generation. Our main results demonstrate computational challenges in filtering harmful input prompts, when there is computational asymmetry between the filters and the LLMs. First, we show that there exist LLMs for which there are no efficient input prompt filters: adversarial prompts, which are provably computationally indistinguishable from benign prompts for any efficient filter but elicit harmful behavior from LLMs, can be easily constructed. Our second main result identifies a natural setting in which output filtering is computationally intractable. We conclude that safety cannot be achieved by designing filters external to the LLM internals (architecture and weights); in particular, black-box access to the LLM will not suffice.

1 INTRODUCTION

Artificial Intelligence (AI) systems, particularly LLMs, are being adopted across a wide array of domains, including business, healthcare, education, and even governance (Potash et al., 2015; Chiusi et al., 2020; Levy et al., 2021; Haensch et al., 2023; Perdomo et al., 2023; Fischer-Abaigar et al., 2024). As the influence of AI expands, ensuring the alignment of these systems with human values has become a critical societal concern. Governments and regulatory bodies around the globe are responding to this challenge by introducing frameworks to classify, monitor, and audit AI systems. For instance, the European Union's AI Act (EU AI Act) mandates extensive risk assessments and management for high-risk AI applications.

Informally, alignment refers to the process of ensuring that a model generates outputs that are consistent with human preferences, essentially teaching the model to generate responses that align with what humans consider safe and beneficial (Amodei et al., 2016; Leike et al., 2018; Bai et al., 2022). In practice, existing alignment mechanisms face significant challenges, as demonstrated by the prevalence of "jailbreak" attacks that successfully bypass model alignment and external safety filters (see empirical results in Section 4 and Jin et al., 2024; Yi et al., 2024; Huang et al., 2025). These empirical challenges raise a fundamental question: can we hope to guarantee the safety of advanced AI systems, or are there intrinsic barriers to such guarantees?

*Authors listed in alphabetical order.

In this paper, we investigate fundamental limitations to achieving AI alignment through the approach of *filtering*. Because many generative-AI models are proprietary and cannot be independently audited and thus trusted, filter-based alignment is an important subject of study. Under standard cryptographic assumptions, we show that significant computational barriers exist to filtering both input prompts and output responses of LLMs in order to prevent harmful content.

Our concrete contributions are as follows:

- **Input filters.** We show a general method for generating jailbreaks, crafted to elicit harmful behavior from LLMs, which provably cannot be distinguished from benign prompts by *input-prompt filters* which run significantly faster than the LLMs. Our results are based on the cryptographic assumption that *time-lock* puzzles exist (Rivest et al., 1996) and hold for any definition of "harmful behavior" and "benign" prompts.
- **Output filters.** Using similar methods, we prove that *for any efficient output filter*, distinguishing between harmful and benign LLM output is impossible, even when the runtime of the filter is larger than that of the LLM. This implies that no *external* (black-box) alignment mechanism will work in the worst case.
- **Mitigation filters.** We then formalize and analyze relaxed mitigation strategies in which filters are allowed to modify prompts or outputs, rather than simply rejecting them. Although these *mitigation filters* have greater expressive power, we show that they, too, require fundamental computational costs, indicating that even these more flexible approaches are subject to inherent limitations.
- **Strong properties of our jailbreaks.** Ideal jailbreaks are (1) *harmful*, (2) *provably indistinguishable from benign prompts*, and (3) *do not require collusion between the attacker and the LLM*. Compared to previous works, our result is the first one to satisfy all three properties at the same time (under cryptographic assumptions).
- **Novel method.** We introduce a novel method for achieving indistinguishability between any notion of benign prompts and harmful prompts. Our method samples benign-looking (yet harmful) prompts using a pseudo-random sampler, which is built from a cryptographic time-lock (Rivest et al., 1996) puzzle that in itself conceals harmful prompts. The LLM can unlock the harmful prompts, whereas the time-constrained filter cannot even distinguish between benign prompts and the constructed harmful prompts.
- **Experiments.** Experimentally, an approach following our theory indeed evades weak filters and, at the same time, elicits harmful behavior from production LLMs, illustrating the practicality of our theoretical modeling.

Taken together, our results reveal deep challenges in achieving provably aligned AI systems via standard filtering techniques. These findings underscore the need for a more rigorous understanding of the computational hardness underlying alignment—and raise important implications for the design of regulatory frameworks and technical safeguards, as one implication of our work is that the government (an auditing body) *should* have access to LLMs' internals, e.g., weights.

2 Input-Prompt Filtering

The first approach we consider is running user prompts through an *input-prompt filter*, which can detect and discard prompts that might lead to harmful answers from the LLM. We focus on a setting in which the LLM itself is separate from the input-prompt filter, but we can query it while training an input-prompt filter (during training we can try different prompts to observe if and where harmful behaviors occur). After this filter-training phase, the filter should detect (and reject) prompts that lead to harmful outputs. This is a natural approach which arises in practice. We ask: *Can efficient input-detection filters always be created?*

Setup: The setup is as follows. There is a computationally powerful unsafe LLM. Namely, there is a non-empty set of prompts that incite the LLM to produce harmful outputs, e.g., requesting instructions on how to build a bomb is considered harmful. Second, there is an input-prompt filter which on getting a prompt from a user, should reject it or pass it to the LLM. The filter is to reject prompts on which the LLM might produce a harmful output. The filter was trained using data and black-box access to the LLM, but during deployment does not have access to the LLM weights. The

filter is only a moderately powerful algorithm, which is more efficient than the LLM - we believe this is an accurate reflection of the current reality where the amount of resources dedicated to capability significantly outweigh the resources dedicated to safety. Finally, there exists an innocent prompt generator G which generates prompts that (when fed into the LLM) produces benign(or innocent) prompt. We assume that the distribution of benign prompts generated by G is not low-entropy (i.e. there are many benign prompts, see Remark 1). One may think of innocent prompts as generated by an honest user (or a subset of its questions that are innocent and contain enough entropy). In practice, it is feasible for an auto-regressive model to imitate such users. To avoid triviality (i.e. having a filter that rejects all prompts), we also require that the filter accepts prompts generated by the innocent prompt generator G with high probability.

Assumption: Our result relies on the existence of cryptographic *Time-Lock Puzzles* and *one-way functions*. Introduced by Rivest et al. (1996) (RSW), time-lock puzzles provide a way to send messages "to the future." A sender can **quickly** generate a puzzle whose solution s remains hidden until a **moderately large amount of computation**—quantified by a time parameter t—has been performed after which it can be solved (or opened). The key property is that s remains inaccessible to any adversary running in significantly less time than t. An extra property that we need is that the puzzle generated is indistinguishable from random for all algorithms which run in a moderately large amount of time, we call this a *pseudo-random time-lock puzzles (PRTLP)*. It is easy to transform the output of a time-lock puzzle, into a pseudo-random puzzle under the existence of one-way functions. Rivest et al. (1996) proposed a concrete construction based on the assumption that repeated squaring modulo a special RSA integer is an *inherently sequential* task. The RSA integer, standing for Rivest, Shamir, and Adleman, is a product of two large primes, and appears in the context of public-key cryptography (Rivest et al., 1978). Other proposals of time-lock puzzles based on a variety of hard problems appear in Bitansky et al. (2016a) (Agrawal et al., 2025; Bitansky and Garg, 2025; Abram et al., 2025).

Idea: Informally, in our setting, we use pseudo-random time-lock puzzles to conceal (lock) harmful instructions and use the resulting puzzle as the randomness of a process (see Section 7) that samples benign prompts. An LLM that runs for time t can recover the randomness, solve (unlock) the puzzle, and follow the harmful instructions, while a filter incapable of spending that much time cannot even distinguish between true randomness and the pseudo-random puzzle. Our main result shows the impossibility of input-prompt filtering as follows.

Theorem 1 (Input-prompt detection filter impossibility (informal))**.** *For any high-entropy innocent prompt generator G, there exists an LLM M' and an adversarial prompt generator G' (with comparable runtime to G) such that: (1) prompts by G' induce M' to produce harmful outputs, and (2) under the assumption that cryptographic Time-Lock Puzzles exist, no input-prompt filter that runs significantly faster than M' can distinguish prompts generated by G from prompts generated by G'.*

2.1 Discussion

A few remarks are in order. First, our result depends on a computational asymmetry between the filter and the LLM: the filter is strictly weaker. We remark that in practice most filter (guard) models are lightweight LLMs with far fewer parameters than the main deployed models. For example, the largest and newest Llama Guard 4 model has around 12B parameters, whereas models like GPT-5 or Gemini are significantly larger. This is partly because very large guard models would introduce substantial latency and cost, making them undesirable. Generally, when designing any AI system, one needs to decide on the balance between the amount of resources devoted to capability and the resources devoted to alignment. Our result implies that filtering for alignment can not be done (in a black-box manner) using much fewer resources than used for capability (LLM). We believe this to be an important take away message, given the possible harm that can be caused by unaligned LLMs especially as we may be heading towards AGI.

Second, as prompt-filtering may be imposed by an entity, which is external to the LLM and does not have full access to the LLM internals (weights, etc), filtering should work for *all LLMs* independent of their internals. Thus, to show the impossibility of filtering, we need only to show that *there exist* LLMs that cannot be aligned using prompt-filtering alone. Indeed, our theorem shows the existence of LLMs for which filtering fails.

The LLM we construct can be triggered by *any* user: the attack does not require collusion, coordination, or a shared secret between the user and the model. Although the behavior of M' is stylized—it enters a special mode in which it recovers randomness, solves a time-lock puzzle, and executes the hidden instruction—recent empirical work suggests that qualitatively similar context-dependent harmful modes can emerge in current-generation LLMs (Hubinger et al., 2019; Greenblatt et al., 2024). For example, models may condition their behavior on whether they appear to be in training or deployment, selectively complying with alignment objectives only in certain regimes. These empirical findings do not instantiate our construction, but they support the relevance of modeling unusual harmful modes that can be activated by ordinary users.

3 Output Filtering

We also consider an alternative approach to alignment: instead of filtering prompts, the defense mechanism utilizes an *output filter* to detect harmful *answers* produced by the LLM. In this setup, a prompt is given to the LLM, which generates a response; both the prompt and the response are then passed to the output filter, which may choose to detect and reject the output. The goal of the output filter is twofold: (1) as with an input-prompt filter, it should reject harmful prompts; and (2) it should reject cases where *the answer is harmful*, even if the prompt appears benign—e.g., when a seemingly innocent prompt elicits a detailed recipe for constructing a bomb.

Output filters may be easier to design than input-prompt filters, since they operate with more information. This motivates the question: *Can efficient output-detection filters always be created (even when input-prompt filters are impossible)?* Again, we answer this question in the negative.

Note that if it is easy to determine whether a given LLM output is harmful, then output filters are trivial to construct. Thus, output filtering becomes meaningful only in the more challenging setting, where assessing the potential harm of a response is itself difficult. This concern becomes more pressing as AI systems grow more powerful, making it increasingly hard to evaluate the downstream consequences of their outputs (Scheurer et al., 2023; Moix et al., 2025; Tang et al., 2025). For example, imagine an AI generating a complex business plan aimed at reducing costs by subtly recommending the exclusion of clients with certain characteristics.

Setup for Output Filtering. Firstly, there is an output filter that receives as input a prompt-answer pair and decides if it should be rejected as harmful. In contrast with the input-filter case, the output filter can be more computationally powerful than the LLM, and thus the result is stronger. Secondly, there exists a harmfulness measure that measures the level of harm of an output in the long term (e.g., the measure can be thought of as the environment or nature "evaluating" the output). Neither of the parties has direct access to this measure, and the measure can "run" in significantly larger time than the LLM or the output filter.

We show the following impossibility result:

Theorem 2 (Output-detection filter impossibility (informal))**.** *For every LLM M and every prompt x there exists an LLM M' and a harmfulness measure $\mathcal{H}'$ such that: (1) for every harmful prompt m, outputs of $M'(m)$ are judged as harmful by $\mathcal{H}'$, (2) under the assumption that Time-Lock Puzzles exist, no efficient output filter (even one that runs for more time than the LLM itself) can distinguish the outputs of $M'(m)$ (for every m) from outputs of $M(x)$, and (3) M' runtime is similar to the runtime of M.*

4 Experiments

To showcase the vulnerability of the filtering approach and to validate our theoretical findings (Theorems 1 and 2), we conducted some small-scale experiments evaluating the effectiveness of two real-world deployed safety filters: Llama Guard 2 8B (Llama Team, 2024) and Shield Gemma 2B (Zeng et al., 2024), across three LLM families of varying sizes (see Table 1)[1]. The experimental pipeline of the first experiment consists of three stages: (1) The filter filters incoming jailbreak prompts, (2) the LLM processes prompts that bypassed the input-prompt filter, (3) the same filter filters the LLM output. We measure two key performance metrics: (1) **Input filter bypass rate:**

[1]We use fine-tuned versions of models that answer harmful questions when unfiltered.

Percentage of jailbreak prompts that successfully bypass the input filter, (2) **Output filter bypass rate:** Of the prompts that bypassed the input filter, the percentage of resulting LLM outputs that also bypass the output filter. We evaluate 1,097 jailbreak prompts from JailbreakBench (Chao et al., 2024b). Table 1 shows results across all model–filter pairs. A sizable fraction of jailbreaks bypass both input and output filters, indicating current filters fail to block them, supporting the relevance of our theoretical modeling.

Table 1: Safety filter bypass rates across language models

Model	Filter	Input filter bypass rate	Output filter bypass rate
Llama 3.2 3B (Teknium et al., 2024)	Llama Guard	34.37%	58.89%
	Shield Gemma	79.49%	83.49%
Qwen 7B (Hartford et al., 2024a)	Llama Guard	34.37%	61.27%
	Shield Gemma	79.49%	85.89%
Mistral 24B (Hartford et al., 2024b)	Llama Guard	34.37%	67.37%
	Shield Gemma	79.49%	89.91%

Based on Fairoze et al. (2025) and leveraging our theoretical insights, we further show that the input filter bypass rate can be increased (Table 2) if we encrypt the jailbreak prompts and simulate a timed-release attack, where the model is first asked to decode the encrypted text and only after the decoding, it is tasked to reply to the decrypted text (see details Appendix H). This timed-release attack is inspired by the time-lock puzzles we describe (Rivest et al., 1996), which Fairoze et al. (2025) translated to the LLM setting. The increase in filter bypass rates via this approach showcases how one can exploit the resource asymmetry between the input filter and the model by hiding harmful prompts under some amount of computation, which results in successful attacks.

Table 2: Input filter bypass rates by guard and encoding type

Guard	No encryption	ASCII	Caesar	Morse	Substitution	Unicode
Llama Guard	34.37%	66.18%	88.06%	81.40%	81.49%	78.40%
ShieldGemma	79.49%	78.40%	91.89%	87.15 %	91.98%	87.97%

To further validate our findings, we test the timed-release attack (Fairoze et al., 2025) with Caesar cipher encryption (shift-3) against Gemini 2.5 Flash (Google DeepMind, 2025) as a frontier downstream and aligned model. We provide two examples (see Appendix H.1) that showcase how the described timed-release attack can improve the jailbreak success over just using the non-encrypted, direct jailbreak in models that have decoding capabilities.

The dual vulnerability—circumventing external defenses while compromising model alignment—provides empirical support for the real-life relevance of our theoretical modeling and illustrates that alignment cannot be achieved solely through external filtering when computational limits exist.

5 Mitigation

Having established computational barriers for both input-prompt filters and output filters, we now examine a more permissive mechanism, which we term a *prompt–mitigation filter*. Here, the filter can modify the prompt and pass it through to the LLM in addition to rejecting it. Mitigation gives the filter more power, which makes the goal of filtering potentially easier to achieve. We show a connection between the security of watermarking schemes and the impossibility of mitigation filtering.

Prompt mitigation and watermarking. A watermarking scheme W lets an LLM creator prove that an output came from their model, even after adversarial post-processing. Watermarking resistant to "all" edits remains beyond the current state of the art: one typically demands that the adversary preserve *some* semantic content—otherwise it could simply delete the text (and with it the watermark). We therefore consider watermarking against adversaries that apply edits from a permissible class E and run in time t. The watermark should remain *indistinguishable* to any such time-t adversary. Our focus is on auto-regressive models, which generate text token by token, and on watermarking schemes that embed the mark by perturbing the model's sampling randomness—a strategy explored by several recent proposals (Kirchenbauer et al., 2023; Kuditipudi et al., 2023; Christ et al., 2023; Golowich and Moitra, 2024). We show that:

Theorem 3 (Impossibility of mitigation-filters (informal))**.** *Let W be a watermarking scheme as above that is resilient to edits from a class E. For any high-entropy, innocent prompt generator G, there exists an adversarial prompt generator G' (with comparable runtime to G) and an LLM M' such that G' generates prompts that will induce harmful outputs from M' even when G''s outputs pass through an efficient prompt-mitigation filter using edits from the class E.*

6 Related Work

Alignment. Making AI models aligned with human preferences is a central concern of contemporary AI research (Amodei et al., 2016; Leike et al., 2018; Hendrycks et al., 2021; Ji et al., 2023). However, a growing body of work suggests that achieving robust alignment is profoundly difficult: Researchers have highlighted issues ranging from the inherent ambiguity in specifying human preferences (Gabriel, 2020; Zhi-Xuan et al., 2024; Sorensen et al., 2024), to problems like shallow alignment induced by properties of the alignment algorithms (Jain et al., 2023; Kotha et al., 2023; Lee et al., 2024) and the alignment data (Qi et al., 2024). The difficulty in robustly aligning models at a deep representational level underscores the need for complementary external mechanisms like filters to detect or prevent harmful model outputs. This is in line with regulatory frameworks such as the EU AI Act, which requires AI systems in the high-risk category to implement an effective risk management system (see Article 9 EU AI Act).

Filters. In response to the need for safer AI systems, practical filtering mechanisms have been developed and deployed. For instance, model developers like Meta have introduced tools such as Llama Guard, designed to classify content as safe or unsafe (Inan et al., 2023). Similarly, cloud service providers like Microsoft Azure offer content filtering capabilities within their AI service implementations (Microsoft Corporation, 2025), and companies like Nvidia also provide solutions aimed at moderating AI-generated content (NVIDIA Corporation, 2025). These approaches represent an ongoing evolution, with classifiers and filters becoming increasingly sophisticated. However, the development of jailbreaks poses a consistent challenge as they are able to bypass filters and internal model alignment (Andriushchenko et al., 2024; Chao et al., 2024a; Xu et al., 2024; Huang et al., 2025). Against the background of this dynamic co-evolution of attack and defense, our work explores the computational intractability of filtering approaches under cryptographic assumptions.

Time Lock Puzzles. It is usually desired that cryptographic schemes cannot be broken by any adversary. An exception is the notion of cryptographic puzzles that can be solved in some preset amount of time (or space) but not faster. Examples of such puzzles (Dwork and Naor, 1992; Rivest et al., 1996) have been used as a way to combat spam or send messages into the future, forcing the spammer (or the future reader of messages) to invest the preset amount of time. The notion of time-lock puzzles introduced by Rivest et al. (1996) following May's time-released cryptography (May, 1993) is especially intriguing in that it allows a user to quickly encrypt a message in such that it can be read only after a longer but set number of time steps. Informally, the sender generates a puzzle with a solution s that remains hidden from adversaries that run in time significantly less than t, including parallel adversaries with polynomially many processors. The original (Rivest et al., 1996) candidate was based on the assumption that exponentiation modulo an RSA integer is an "inherently sequential" computation. More recently, Bitansky et al. (2016a) constructed time-lock puzzles based on the necessary assumption that worst-case Non-Parallelizing Languages exist—decidable in time t, but hard in the worst case for circuits of depth significantly smaller than t—and that indistinguishability obfuscation exists (Bitansky et al., 2015; Canetti et al., 2014). A culmination of a series of follow-up works (Agrawal et al., 2025; Bitansky and Garg, 2025; Abram et al., 2025) managed to construct time-lock puzzles based on a version of the Learning With Errors (Regev, 2009) (LWE) assumption. One advantage of basing time-lock puzzles on LWE is that the other constructions are broken by quantum algorithms, while LWE is a candidate for post-quantum secure cryptography.

Jailbreaks and encryption. In the following we provide a comparison of papers that are closely related to our paper. Yuan et al. (2024) use a purely empirical approach to demonstrate that current LLMs can be jailbroken using encryption (e.g., Caesar, Morse, ASCII), where the entire conversation occurs in the cipher. This leads them to conclude that cipher data should be included in the safety training as a defense. In contrast, the main contribution of our work provides formal impossibility results: we prove under cryptographic assumptions that filtering-based alignment faces fundamental

computational barriers that cannot be overcome through better training data. Our empirical setup also differs: we use time-lock puzzles where the model decodes encrypted jailbreaks and responds in natural language.

Both Glukhov et al. (2024) and Theorem 2 study barriers to detecting harmful LLM outputs when harmfulness may only become apparent to a downstream user. Their examples rely either on undecidability, or on an encoded output that can be decoded only by a user who shares a secret with the model. In contrast, our time-lock construction requires no shared secret or collusion: it is possible to "lock" the harmful content behind a prescribed amount of computation, and any downstream user willing to spend this computation can unlock it, while a faster output filter cannot. Moreover, Glukhov et al. (2024) propose Mosaic prompts, which split a harmful request into several benign-looking sub-prompts whose answers can later be combined. Our construction instead gives a provable way to hide a harmful prompt or output inside a single benign-looking object, using time-locked steganography. Finally, while Glukhov et al. (2024) discuss syntactic filters for suspicious-looking outputs in the appendix, our construction rules out such filters in the worst case: the harmful content can be hidden "in plain sight" while remaining computationally inaccessible to the filter.

Finally, a new follow-up work (Fairoze et al., 2025) designs an attack that successfully jailbreaks models that are designed to be aligned (where the alignment mechanism is embedded inside the models), e.g., Google Gemini (2.5 Flash/Pro), DeepSeek Chat (DeepThink), Grok (3), and Mistral Le Chat (Magistral). Their attack is explicitly inspired by our time-lock idea to hide harmful prompts under some amount of computation. It shows that the alignment mechanisms embedded inside production models are not able to detect harmful commands hidden with a time-lock-like mechanism, but the models can eventually recover the commands and produce harmful outputs.

7 Technical Overview

Preliminaries. For $n \in \mathbb{N}$ we denote $\{0, 1, \ldots, n-1\}$ by $[n]$ or $\mathbb{Z}_n$. A language model M is a deterministic algorithm that takes as input a prompt and a previous output by the model $z = (z_0, \ldots, z_{i-1})$ and outputs a probability distribution over a token set $\mathcal{T}$. To sample a response of M, one repetitively samples from the probability distribution output by the model.

7.1 Construction

Pseudorandom time-lock puzzle (PRTLP). As we explained, a key component in the construction for Theorem 1 is a PRTLP. A creator of the puzzle can quickly generate a puzzle whose solution s remains hidden until a moderately large amount of computation—quantified by a time parameter t—has been performed, after which it can be solved (or opened). Solution s remains inaccessible to any adversary running in significantly less time than t, and the puzzle is indistinguishable from random for all such adversaries.

We construct a PRTLP based on the RSW time-lock puzzle (Rivest et al., 1996). It relies on the moderately hard repeated exponentiation function:

$$f(n, r) = r^{(2^t)} \pmod{n},$$

where n is a product of two (random) primes p and q, i.e., $n = pq$, r is a random integer between 0 and $n-1$ and t is a fixed large integer. The assumption is that without knowing the factorization of n, of which we think of as a trapdoor, computing f takes time that scales linearly with t (and moreover this computation cannot be parallelized). It is easy to generate (PUZZLE, SOLUTION) pairs: the creator of the puzzle samples n, which is why they know the factorization $n = pq$. It can therefore compute $e = 2^t \pmod{\phi(n)}$[2] where we know that

$$r^e = r^{2^t} \pmod{n},$$

which implies that $\log(e)$, which is not larger than $\log(n)$, exponentiations are enough for puzzle generation. Summarizing, PUZZLE $= (r, n)$ and SOLUTION $= f(n, r) = r^{(2^t)}$. See Figure 1 for a visual representation.

[2] $\phi(n) = (p-1)(q-1)$ is the Euler's totient function.

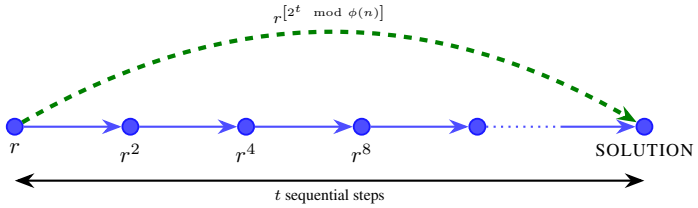

Figure 1: RSW Time-Lock Puzzle: The blue path represents sequential squaring operations requiring t steps, while the green dashed line shows the trapdoor shortcut.

However, $\text{PUZZLE} = (r, n)$ is not random, e.g., n is a product of two primes. Later (Section 7.2), we will develop a procedure $\mathsf{PseduoRand}(r, n)$ that makes the puzzle look pseudorandom, i.e., indistinguishable from random for adversaries running significantly faster than t. PseudoRand will have an associated algorithm Sol that solves the puzzle. We note that the exact computational gap can be quantified and is usually in the form t vs t^ϵ for some $\epsilon \in (0, 1)$ (see Definition 4).

Recoverable-Randomness Sampling (RRS). We model benign prompts as generated by an auto-regressive model G (see Appendix C for a formal definition). We think of G as a formalization of a subset of prompts the input filter should accept. G is known to all parties.[3]

We design a scheme called Recoverable-Randomness Sampling (RRS), which consists of a pair of algorithms Samp (Algorithms 1) and RecoverRand (Algorithms 2), where:

- $\mathsf{Samp}_G(R) \to z$; takes as input randomness R and outputs a prompt z.
- $\mathsf{RecoverRand}_G(z) \to R$; on input z recovers randomness R.

Samp_G samples from a distribution that is *equal* to that of G, but the way it uses its randomness is different so that it can be recovered by $\mathsf{RecoverRand}_G$. We remark that this is nontrivial to realize. For example, imagine G samples a particular token with probability $3/4$. To do it, G needs 2 bits of randomness but upon seeing this token, it is impossible to know which of the 3 possible choices of randomness was used. Our construction of an RRS, given in Section 7.3, was inspired by ideas from the watermarking litereature (e.g., Christ et al., 2023).

Hiding jailbreaks in plain sight. Assume M is an LLM for which there exist many harm-eliciting prompts HARM, e.g., "How to build a bomb?". We design an adversarial prompt generator G' and an LLM M' such that $G'(\text{HARM})$ produces a prompt z that: (1) elicits harmful behavior from M', and (2) is indistinguishable from prompts from G, for filters running faster than t .

$G'(\text{HARM})$ works as follows:

1. $((r, n), \text{SOLUTION}) \leftarrow \mathsf{RSWTimeLock}(t)$.
2. $\text{PUZZLE} \leftarrow \mathsf{PseudoRand}(r, n)$.
3. $R \leftarrow (\text{PUZZLE}, \mathsf{Enc}(\text{SOLUTION}, \text{HARM}))$, i.e., in Section 7.2 we define a pseudorandom encryption that allows encrypting HARM using SOLUTION as key into a pseudorandom ciphertext.
4. Return $z \leftarrow \mathsf{Samp}_G(R)$.

$M'(z)$ works as follows:

1. Recover $R \leftarrow \mathsf{RecoverRand}_G(z)$.
2. Parse R as (PUZZLE, c)
3. $\text{SOLUTION} = \mathsf{Sol}(\text{PUZZLE})$.
4. Decrypt $c = \mathsf{Enc}(\text{SOLUTION}, \text{HARM})$ using SOLUTION as the decryption key to obtain HARM.

[3]We emphasize that G is neither a secret key nor a public key chosen (published) by the LLM.

5. If recovery was succesful[4] return $M(\text{HARM})$, and return $M(z)$ otherwise.

This construction satisfies the following two properties that essentially imply Theorem 1 automatically:

- The filter cannot distinguish z from a benign prompt from G, since R is indistinguishable from random for algorithms running in significantly fewer steps than t.
- M' is able to recover R, compute a solution to the puzzle embeded in R, recover HARM, and execute it.

Formal definitions of G' and M' are given in Algorithm 3, and Algorithm 4, both in Appendix E.1, respectively. We proceed with the construction of PseudoRand.

7.2 Making RSW time-lock pseudorandom

We proceed with a definition of $\mathsf{PseduoRand}(r, n)$. The previous discussion was simplified in a few points we discuss now. First, the composite n is not a random string (which is what we need), rather it is a product of two large primes. However, we know that n is uniformly random in the set of products of two primes p and q chosen at random from a set $[1, w]$. By the prime number theorem, the number of primes in that range is $\Theta(w/\log(w))$. Thus, n is uniformly random in a domain of size $S = \Theta((w/\log(w))^2)$. G' hashes n to a range of size $\Theta(\epsilon^2 \cdot S)$ using a pairwise independent hash function h. By the leftover hash lemma the joint distribution of $(h, h(n))$ is ϵ-close to uniformly random (Håstad et al., 1999; Vadhan, 2012). It is straightforward to represent the random integer r as a random bit string, so the entire input $x = (h, h(n), r)$ can be (close to) uniformly random, as we wanted. We remark that the LLM will need to recover n from $(h, h(n))$: this is possible because the pairwise-independent hash function can be inverted efficiently (e.g., it can be a linear function, where inverses can be found using Gaussian elimination). Note however that there will be more than one inverse (because the hash function is shrinking), and this raises technical issues in the construction: the LLM will need to enumerate over all the possibilities. The final issue is that the output $r^{(2^t)} \pmod n$ is moderately hard for the filter to compute, but might not be indistinguishable from uniformly random (which is what we wanted). We handle this using standard techniques from the cryptography literature (namely using hardcore bits of f (Goldreich and Levin, 1989). See the appendix for details of the construction.

7.3 Recoverable-Randomness Sampling

In our construction, it is key that the distribution of G' is indistinguishable from G and that it is possible to effectively recover the randomness used by G' from its outputs. To achieve this, we design an algorithm Samp (Algorithm 1 in Appendix D) that takes as input randomness R and produces $z = (z_0, z_1, \dots)$; and an algorithm RecoverRand (Algorithm 2 in Appendix D) that given z recovers R. We assume that the set of tokens over which G operates equals $\{0, 1\}$ (see Appendix C).

The high-level idea of our scheme is to maintain "residual randomness". Imagine, that at some point in the generation process, $G(z_0, \dots, z_{i-1}) \to 1/4$. Then to generate z_i accurately 2 bits of randomness are needed, i.e., if the bits are 00 we set $z_i = 1$ and we set $z_i = 0$ if the bits are $00, 01, 11$. If $z_i = 0$ then the recovering algorithm might not know which of the three options for the randomness determined z_i. To address this issue we "reuse" this randomness in the generation of the next tokens. More concretely, if $z_i = 0$ then we have $\log_2(3)$ bits of residual randomness that we can use in generating $z_{i+1}, z_{i+2}, \dots$. If done properly this "reusing" process will allow perfect recovery of the randomness used. Intuitively, because all of the randomnness will be used.

Our scheme, defined in Algorithms 1 and 2, works as follows. At all times, the sampling algorithm $\mathsf{Samp}_G(R)$ maintains a precision range k and a value $q \in [k]$. The value q is created on the fly using the randomness R and will determine how tokens are sampled. Intuitively, q maintains the "residual randomness". Samp_G builds a response $(z_0, \dots, z_{i-1})$ iteratively, where in the i-th step it:

[4]The encryption scheme we design has an additional feature of detecting if decryption was successful. If z was sampled from G, then the probability of successful decryption would be very small. This implies that the harmful "mode" of M' is almost never triggered when given prompts from G.

1. Calls $p_i \leftarrow G(z_0, \ldots, z_{i-1})$.
2. If $k < 2^P$ it multiplies k by an appropriate power 2^t so that $k \geq 2^P$,[5] and increases the precision of q by concatenating it with the not yet used randomness from R, i.e., $q \leftarrow q \parallel R[j : j + t]$. This ensures that q represents an element of $[k]$.
3. If $q/k < p_i$ it sets $z_i \leftarrow 1$ and sets $z_i \leftarrow 0$ otherwise. Additionally, it updates q and k so that the "residual randomness" can be used later on. Intuitively, if $q/k < p_i$ then we keep the $\approx \log_2(p_i \cdot k)$ bits of randomness. To do that we update $k \leftarrow k - \lfloor p_i \cdot k \rfloor$, which implicitly represents that q is now a random value on $[k - \lfloor p_i \cdot k \rfloor]$. If $q/k > p_i$, the values are updated accordingly.

To recover the randomness $\mathsf{RecoverRand}_G(z)$ works as follows. It maintains bitstrings a, b that informally are "lower and upper-bounds" on the randomness R being recovered. More concretely, in the first step RecoverRand knows that the prefix of randomness R is between 0^P and 1^P. If $p_0 \leftarrow G()$ and $z_0 = 1$ then it knows that the prefix of R is between $a = 0^P$ and $b = p_0 \cdot 2^P$. By construction, the common prefix of a, b is equal to the prefix of R - this will allow to recover R. Observe that it is possible to update k exactly like Samp_G does it because k changes depending only on the value of z_i.

7.4 Output Filtering

Setup. A *harmfulness oracle* is a function $\mathcal{H} : \mathcal{T}^* \to [0, \infty)$, where $\mathcal{H}(z) = 0$ means that output z is benign and larger values indicate greater harm. During inference, neither the LLM nor the filter can query $\mathcal{H}$; rather, $\mathcal{H}$ models the downstream environment evaluating the eventual harm of z. The LLM and filter may nevertheless have seen training examples or an approximate description of $\mathcal{H}$.

Construction. We leverage techniques used in the proof of Theorem 1 to prove Theorem 2. Let M be an LLM that doesn't produce harmful outputs.

To define M' we apply the same process as the one used to transform $G \dashrightarrow G'$ but applied to M. When we apply the transformation, we set t of the RSW time-lock to be exponential in the time complexity of M. This ensures that the outcome of the transformation, i.e., M', is such that its outputs are indistinguishable from outputs of M for all polynomial, in the running time of M, filters. $\mathcal{H}'(z)$ "runs" as follows: it recovers the randomness used by M, i.e., $R \leftarrow \mathsf{RecoverRand}_M(z)$. If R contains a time-lock puzzle, $\mathcal{H}'$ solves it to obtain an output m and "declares" that z is harmful (with harmfulness level determined by m). The proof of correctness is similar to the proof of Theorem 1.

8 Conclusions

Many generative AI models are proprietary and cannot be independently audited, making filter-based alignment a critical area of study. We therefore conduct a theoretical investigation into the computational requirements for achieving AI alignment through *filtering* approaches.

Under cryptographic assumptions, we prove that input filters substantially weaker (computationally) than the LLM cannot successfully prevent jailbreaking. This conclusion extends beyond the detection of harmfulness to a class of filters which actively change prompts in order to mitigate their effect. Similarly, output filters cannot successfully prevent harmful outputs if they are computationally weaker than the downstream environment (i.e., the end user of the LLM output). Ultimately, the only way to judge the level of harm of an LLM output is to execute it in the world.

We present experiments to demonstrate that time-lock inspired attacks evade weak filters and elicit harmful behavior from production LLMs. To achieve our results, we introduce a novel method to craft harmful prompts so they are indistinguishable from benign prompts using time-lock cryptographic puzzles—a technique with potentially broader applications.

In conclusion, filtering for alignment requires computational resources comparable to those used for the LLM itself, along with access to the model's internals (architecture and weights). This has an important implication: resources invested in safety must match or exceed those invested in capability. Given the potential harms of unaligned LLMs, particularly as we approach AGI, this resource parity is essential.

[5] P is a precision parameter that determines the closeness of the generated distribution to G.

ACKNOWLEDGEMENTS

This work is in part supported by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the German Federal Ministry of Education and Research (SB), and was done in part while SB and FK were visiting the Simons Institute for the Theory of Computing. OR is supported by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness, the Sloan Foundation Grant 2020-13941, and the Simons Foundation Investigators Award 689988.

REPRODUCIBILITY STATEMENT

Full proofs for all theoretical results are provided in the Appendix. Details for implementing the experiments are given in the text and the appendix. The code is available on GitHub.

REFERENCES

D. Abram, G. Malavolta, and L. Roy. Key-homomorphic computations for RAM: fully succinct randomised encodings and more. *IACR Cryptol. ePrint Arch.*, page 339, 2025. URL https://eprint.iacr.org/2025/339.

S. Agrawal, G. Malavolta, and T. Zhang. Time-lock puzzles from lattices. *IACR Cryptol. ePrint Arch.*, page 47, 2025. URL https://eprint.iacr.org/2025/047.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

M. Andriushchenko, F. Croce, and N. Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

Anthropic. Claude 4, 2025. URL https://www.anthropic.com/news/claude-4. Large language model.

Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Y. Bengio, M. Cohen, D. Fornasiere, J. Ghosn, P. Greiner, M. MacDermott, S. Mindermann, A. Oberman, J. Richardson, O. Richardson, M.-A. Rondeau, P.-L. St-Charles, and D. Williams-King. Superintelligent agents pose catastrophic risks: Can scientist ai offer a safer path?, 2025. URL https://arxiv.org/abs/2502.15657.

N. Bitansky and R. Garg. Succinct randomized encodings from laconic function evaluation, faster and simpler. In *Advances in Cryptology – EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4–8, 2025, Proceedings, Part VII*, page 406–436, Berlin, Heidelberg, 2025. Springer-Verlag. ISBN 978-3-031-91097-5. doi: 10.1007/978-3-031-91098-2_15. URL https://doi.org/10.1007/978-3-031-91098-2_15.

N. Bitansky, S. Garg, H. Lin, R. Pass, and S. Telang. Succinct randomized encodings and their applications. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 439–448, 2015.

N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 345–356, 2016a.

N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, page 345–356, New York, NY, USA, 2016b. Association for Computing Machinery. ISBN 9781450340571. doi: 10.1145/2840728.2840745. URL https://doi.org/10.1145/2840728.2840745.

D. Boneh and M. Naor. Timed commitments. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, page 236–254, Berlin, Heidelberg, 2000. Springer-Verlag. ISBN 3540679073.

D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018. doi: 10.1007/978-3-319-96884-1_25. URL https://doi.org/10.1007/978-3-319-96884-1_25.

R. Canetti, J. Holmgren, A. Jain, and V. Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. *Cryptology ePrint Archive*, 2014.

P. Chao, E. Debenedetti, A. Robey, M. Andriushchenko, F. Croce, V. Sehwag, E. Dobriban, N. Flammarion, G. J. Pappas, F. Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*, 2024a.

P. Chao, E. Debenedetti, A. Robey, M. Andriushchenko, F. Croce, V. Sehwag, E. Dobriban, N. Flammarion, G. J. Pappas, F. Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37: 55005–55029, 2024b.

F. Chiusi, B. Alfter, M. Ruckenstein, and T. Lehtiniemi. Automating society report 2020. 2020.

M. Christ, S. Gunn, and O. Zamir. Undetectable watermarks for language models. *IACR Cryptol. ePrint Arch.*, 2023:763, 2023. URL https://api.semanticscholar.org/CorpusID:259092330.

W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi: 10.1109/TIT.1976.1055638.

C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Annual international cryptology conference*, pages 139–147. Springer, 1992.

M. J. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. E. Bassham, E. Roback, and J. D. Jr. Advanced encryption standard (aes), 2001-11-26 00:11:00 2001. URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901427.

EU AI Act. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence. *OJ*, L 2024/1689.

J. Fairoze, S. Garg, K. Lee, and M. Wang. Bypassing prompt guards in production with controlled-release prompting, 2025. URL https://arxiv.org/abs/2510.01529.

U. Fischer-Abaigar, C. Kern, N. Barda, and F. Kreuter. Bridging the gap: Towards an expanded toolkit for ai-driven decision-making in the public sector. *Government Information Quarterly*, 41 (4):101976, 2024.

I. Gabriel. Artificial intelligence, values, and alignment. *Minds and machines*, 30(3):411–437, 2020.

D. Glukhov, I. Shumailov, Y. Gal, N. Papernot, and V. Papyan. Position: Fundamental limitations of LLM censorship necessitate new approaches. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=j5csKrtyAe.

O. Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, USA, 2006. ISBN 0521035368.

O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In D. S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 25–32. ACM, 1989. doi: 10.1145/73007.73010. URL https://doi.org/10.1145/73007.73010.

N. Golowich and A. Moitra. Edit distance robust watermarks via indexing pseudorandom codes. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 20645–20693. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/24c53bfa5b53fc2cf05644f5a7a26bb0-Paper-Conference.pdf.

Google DeepMind. Gemini 2.5 flash. https://deepmind.google/models/gemini/flash/, 2025. Accessed: 2025-11-21.

R. Greenblatt, C. Denison, B. Wright, F. Roger, M. MacDiarmid, S. Marks, J. Treutlein, T. Belonax, J. Chen, D. Duvenaud, A. Khan, J. Michael, S. Mindermann, E. Perez, L. Petrini, J. Uesato, J. Kaplan, B. Shlegeris, S. Bowman, and E. Hubinger. Alignment faking in large language models, 12 2024.

A.-C. Haensch, S. Ball, M. Herklotz, and F. Kreuter. Seeing chatgpt through students' eyes: An analysis of tiktok data. In *2023 Big Data Meets Survey Science (BigSurv)*, pages 1–8. IEEE, 2023.

E. Hartford, L. Atkins, F. Fernandes, and C. Computations. Dolphin 2.9.2 qwen2 7b. Hugging Face, 2024a. URL https://huggingface.co/dphn/dolphin-2.9.2-qwen2-7b. Fine-tuned model based on Qwen2-7B.

E. Hartford, B. Gitter, BlouseJury, and C. Computations. Dolphin 3.0 mistral 24b. Hugging Face, 2024b. URL https://huggingface.co/dphn/Dolphin3.0-Mistral-24B. Fine-tuned model based on Mistral-Small-24B-Base-2501.

J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi: 10.1137/S0097539793244708. URL https://doi.org/10.1137/S0097539793244708.

D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021.

Y. Huang, N. Bray, A. Rao, Y. Ji, and W. Hu. How good are the llm guardrails on the market? A comparative study on the effectiveness of LLM content filtering across major GenAI platforms, June 2025. URL https://unit42.paloaltonetworks.com/comparing-llm-guardrails-across-genai-platforms/. Accessed: July 23, 2025.

E. Hubinger, C. Merwijk, V. Mikulik, J. Skalse, and S. Garrabrant. Risks from learned optimization in advanced machine learning systems, 06 2019.

H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

S. Jain, R. Kirk, E. S. Lubana, R. P. Dick, H. Tanaka, E. Grefenstette, T. Rocktäschel, and D. S. Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. *arXiv preprint arXiv:2311.12786*, 2023.

J. Ji, T. Qiu, B. Chen, B. Zhang, H. Lou, K. Wang, Y. Duan, Z. He, J. Zhou, Z. Zhang, et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

H. Jin, A. Zhou, J. Menke, and H. Wang. Jailbreaking large language models against moderation guardrails via cipher characters. *Advances in Neural Information Processing Systems*, 37:59408–59435, 2024.

J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

S. Kotha, J. M. Springer, and A. Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. *arXiv preprint arXiv:2309.10105*, 2023.

R. Kuditipudi, J. Thickstun, T. Hashimoto, and P. Liang. Robust distortion-free watermarks for language models. *CoRR*, abs/2307.15593, 2023. doi: 10.48550/ARXIV.2307.15593. URL https://doi.org/10.48550/arXiv.2307.15593.

A. Lee, X. Bai, I. Pres, M. Wattenberg, J. K. Kummerfeld, and R. Mihalcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. *arXiv preprint arXiv:2401.01967*, 2024.

J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, and S. Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

K. Levy, K. E. Chasalow, and S. Riley. Algorithms and decision-making in the public sector. *Annual Review of Law and Social Science*, 17(1):309–334, 2021.

Llama Team. Meta llama guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md, 2024.

T. C. May. Timed-release cryoto. *http://www. hks. net/cpunks/cpunks-0/1460. html*, 1993.

Microsoft Corporation. What is azure ai content safety? https://learn.microsoft.com/en-us/azure/ai-services/content-safety/overview, 2025. Accessed: 2025-05-14.

A. Moix, K. Lebedev, and J. Klein. Threat intelligence report: August 2025. Technical report, Anthropic, August 2025. URL https://www-cdn.anthropic.com/b2a76c6f6992465c09a6f2fce282f6c0cea8c200.pdf. Case studies of AI misuse including vibe hacking, remote worker fraud, and malware development.

NVIDIA Corporation. Nvidia nemo guardrails documentation. https://docs.nvidia.com/nemo/guardrails/latest/index.html, 2025. Accessed: 2025-05-14.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024. URL https://arxiv.org/abs/2303.08774.

J. C. Perdomo, T. Britton, M. Hardt, and R. Abebe. Difficult lessons on social prediction from wisconsin public schools. *arXiv preprint arXiv:2304.06205*, 2023.

E. Potash, J. Brew, A. Loewi, S. Majumdar, A. Reece, J. Walsh, E. Rozier, E. Jorgenson, R. Mansour, and R. Ghani. Predictive modeling for public health: Preventing childhood lead poisoning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2039–2047, 2015.

X. Qi, A. Panda, K. Lyu, X. Ma, S. Roy, A. Beirami, P. Mittal, and P. Henderson. Safety alignment should be made more than just a few tokens deep. *arXiv preprint arXiv:2406.05946*, 2024.

O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), Sept. 2009. ISSN 0004-5411. doi: 10.1145/1568318.1568324. URL https://doi.org/10.1145/1568318.1568324.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978. ISSN 0001-0782. doi: 10.1145/359340.359342. URL https://doi.org/10.1145/359340.359342.

R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. 1996.

J. Scheurer, M. Balesni, and M. Hobbhahn. Large language models can strategically deceive their users when put under pressure. *arXiv preprint arXiv:2311.07590*, 2023.

T. Sorensen, J. Moore, J. Fisher, M. Gordon, N. Mireshghallah, C. M. Rytting, A. Ye, L. Jiang, X. Lu, N. Dziri, et al. A roadmap to pluralistic alignment. *arXiv preprint arXiv:2402.05070*, 2024.

X. Tang, Q. Jin, K. Zhu, T. Yuan, Y. Zhang, W. Zhou, M. Qu, Y. Zhao, J. Tang, Z. Zhang, et al. Risks of ai scientists: prioritizing safeguarding over autonomy. *Nature Communications*, 16(1):8317, 2025.

R. Teknium, J. Quesnelle, and C. Guang. Hermes 3 technical report, 2024. URL https://arxiv.org/abs/2408.11857.

S. P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi: 10.1561/0400000010. URL https://doi.org/10.1561/0400000010.

Z. Xu, Y. Liu, G. Deng, Y. Li, and S. Picek. A comprehensive study of jailbreak attack versus defense for large language models. *arXiv preprint arXiv:2402.13457*, 2024.

S. Yi, Y. Liu, Z. Sun, T. Cong, X. He, J. Song, K. Xu, and Q. Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024.

Y. Yuan, W. Jiao, W. Wang, J. tse Huang, P. He, S. Shi, and Z. Tu. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=MbfAK4s61A.

O. Zamir. Undetectable steganography for language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=fq6aQoMSHz.

W. Zeng, Y. Liu, R. Mullins, L. Peran, J. Fernandez, H. Harkous, K. Narasimhan, D. Proud, P. Kumar, B. Radharapu, O. Sturman, and O. Wahltinez. Shieldgemma: Generative ai content moderation based on gemma, 2024. URL https://arxiv.org/abs/2407.21772.

T. Zhi-Xuan, M. Carroll, M. Franklin, and H. Ashton. Beyond preferences in ai alignment. *Philosophical Studies*, pages 1–51, 2024.