FEBRUARY 2021

AI Verification

Mechanisms to Ensure AI Arms Control Compliance

CSET Issue Brief

CENTER *for* SECURITY *and* EMERGING TECHNOLOGY

AUTHOR
Matthew Mittelsteadt

Executive Summary

The international system is at an artificial intelligence fulcrum point. Compared to humans, AI is often faster, fearless, and efficient. National security agencies and militaries have been quick to explore the adoption of AI as a new tool to improve their security and effectiveness. AI, however, is imperfect. If given control of critical national security systems such as lethal autonomous weapons, buggy, poorly tested, or unethically designed AI could cause great harm and undermine bedrock global norms such as the law of war. To balance the potential harms and benefits of AI, international AI arms control regulations may be necessary.

Proposed regulatory paths forward, however, are diverse. Potential solutions include calls for AI design standards to ensure system safety, bans on more ethically questionable AI applications, such as lethal autonomous weapon systems, and limitations on the types of decisions AI can make, such as the decision to use force. Regardless of the chosen regulatory scheme, however, there is a need to verify an actor's compliance with regulation. AI verification gives teeth to AI regulation.

This report defines "AI Verification" as the process of determining whether countries' AI and AI systems comply with treaty obligations. "AI Verification Mechanisms" are tools that ensure regulatory compliance by discouraging or detecting the illicit *use* of AI by a system or illicit *AI control* over a system.

Despite the importance of AI verification, few practical verification mechanisms have been proposed to support most regulation in consideration. Without proper verification mechanisms, AI arms control will languish. To this end, this report seeks to jumpstart the regulatory conversation by proposing mechanisms of AI verification to support AI arms control.

Due to the breadth of AI and AI arms control policy goals, many approaches to AI verification exist. It is well beyond the scope of this report to focus on all possible options. For the sake of brevity, this report addresses the subcase of verifying whether an AI exists in a system and if so, what functions that AI could command. This report also focuses on mechanical systems, such as military drones, as the target of regulation and verification mechanisms. This reflects the focus of a wide range of regulatory proposals and the policy goals of many organizations fighting for AI arms control. In sum, this report concentrates on verification mechanisms that support many of the most popular AI arms control policy goals. Naturally, other approaches exist and should be studied further, however; they are beyond the scope of this initial

report on the subject. To these ends, this report presents several novel verification mechanisms including:

System Inspection Mechanisms: Mechanisms to verify, through third party inspection, whether any AI exists in a given system and whether that AI could control regulated functions:

- **Verification Zone Inspections:** An inspection methodology that uses limited-scope software inspections to verify that any AI in a system cannot control certain functions. The subsystems that AI must not control, for example, subsystems controlling the use of force, are designated as "verification zones." If these select verification zones can be verified as free from AI control, the system as a whole is compliant. This limited inspection scope reduces the complexity of system inspections, protects subsystems irrelevant to AI regulation, and renders inspections less intrusive.
- **Hardware Inspections:** The existence of AI in subsystems and its control over certain functions can be verified by examining whether AI chips exist and what subsystems they control.

Sustained Verification Mechanisms: These are tools that can be used to verify a system remains compliant after an initial inspection:

- **Preserving Compliant Code with Anti-Tamper Techniques:** These techniques protect system software from post-inspection tampering that may alter what AI can control. Methods chosen to illustrate such techniques include cryptographic hashing of code and code obfuscation. Cryptographically hashed system software also provides a record of expected system design for inspectors that can be used to monitor system software compliance long term.
- **Continuous Verification through Van Eck Radiation Analysis:** Verified systems can be affixed with a Van Eck radiation monitoring mechanism that can be used to monitor the radiation the system produces when code is run. Aberrations detected in this radiation could indicate non-compliant manipulation.

This report introduces and explains why these mechanisms have potential to support an AI verification regime. However, further research is needed to fully assess their technical viability and whether they can be implemented in an operationally practical manner.

Table of Contents

Introduction

Artificial intelligence has emerged as one of the most notable new national security technologies. While diverse in form and function, the common thread that unites the constellation of AI technologies is the unique power to grant autonomy and beyond-human insight to a range of systems and processes. The national security benefits of AI are obvious. One can easily imagine, and in many cases already see, the powerful new intelligence tools and military capabilities driven by AI. In the right hands, AI systems promise invaluable security benefits; in the wrong hands, they could be a grave threat. If given control of critical national security systems such as lethal autonomous weapons, buggy, poorly tested, or unethically designed AI could cause great harm and undermine bedrock global norms such as the law of war. Few argue AI systems be left unconstrained. To balance the potential harms and benefits of AI, international AI arms control regulations may be necessary.

If used, AI systems must be safe, adhere to the law, and have an overall net benefit. To these commonly accepted ends, various policy goals and regulations have been proposed to delimit acceptable design and use of AI systems under international law. Call it "AI arms control." Discourse, however, has sputtered in part due to a lack of practical mechanisms to verify a system's compliance with proposed international regulations.[*]

The standard national security term of art for ensuring compliance with arms control is "verification." The United Nations Institute for Disarmament Research and other United Nations bodies define verification as "the collection, collation and analysis of information in order to make a judgement as to whether a party is complying with its obligations.[1] To address a potential point of confusion, this same term is also used by computer scientists to describe the process of analyzing whether software behaves as expected. In this report, verification is used strictly in the national security sense.

Without verification, international AI arms control lacks teeth. Effective AI arms control may continue to languish unless practical verification mechanisms are developed. This report seeks to jumpstart this discussion by proposing several novel AI verification mechanisms.

[*] See Appendix A for a discussion of the difficulties that have plagued the development of effective AI verification techniques.

In this report, I define "AI Verification" as:

> The processes of determining whether countries' AI and AI systems comply with treaty obligations.

I define "AI Verification Mechanisms" as:

> The set of mechanisms that ensure regulatory compliance by discouraging or detecting the illicit *use* of AI by a system or illicit *AI control* over a system.

As mentioned, AI is a broad class of technologies. Regulators cannot expect a single silver bullet that can eliminate AI threats. Regulations will likely have to be technology specific, and verification will have to use a range of overlapping mechanisms. Recognizing the breadth of this topic and the unwieldy task of addressing all possible regulatory outcomes, this report strives not to answer all questions and solve every problem, but to get the process started.

For the sake of brevity, this report specifically addresses the subcase of verifying whether an AI exists in a system or subsystem and if so, what functions that AI could command. I focus on mechanical systems, such as military drones, as the target of regulation and verification mechanisms. I refer to these mechanical systems as simply "systems" or "AI systems," as appropriate, throughout. These choices reflect the emphasis of a wide range of regulatory proposals and the policy goals of many organizations lobbying for AI arms control. These include regulations mandating a human remains in the loop for use of force decisions, bans on certain systems that should not be AI controlled, and other regulatory proposals that seek to limit *what can use AI and what AI can do.* In sum, this report concentrates on verification mechanisms that support many of the most popular AI arms control policy goals.

Naturally, there are many other routes that can be taken to regulate AI systems. Other possible options include safety and control demonstrations, rules regulating thoroughness of testing, and the ubiquitous implementation of certain norms. These and other options are beyond the scope of this report but should be considered further by interested policymakers and researchers.

This report is intended for those policymakers and national security leaders who oversee AI systems (and systems that may one day become AI systems) or negotiate international policy to control their use. It is important to highlight that the mechanisms contained within this report represent a regulatory starting point. Verification is technically complex, detailed, and politically difficult. Each mechanism should be passed to engineers for further research and to determine how it can be implemented in current or future state systems. This report is a list of possibilities, not answers. Policymakers must lean on their engineers and diplomats to rework and build on these ideas as needed to fit technical and political reality.

A Note on Spoofing

Many verification mechanisms could be subject to spoofing. In the context of AI verification, spoofing can be thought of as any method that an actor may use to deceive regulators and pursue illicit AI activities. It should be assumed that if an actor has the sufficient will and resources to cheat, they could develop spoofing methods to circumvent verification. This does not mean verification is fruitless, merely imperfect. Former Deputy Secretary of Defense Paul Nitze summarized these realities by stating the goal of effective verification is to make sure that:

> [I]f the other side moves beyond the limits of the Treaty in any militarily significant way, we would be able to detect such violation in time to respond effectively and thereby deny the other side the benefit of the violation.[2]

A practical verification mechanism does not necessarily render spoofing impossible but seeks to catch spoofing in time to act. That said, this stated goal comes with the asterisk that even methods failing to guarantee this basic requirement can help. A second goal of verification is discouragement. If a verification mechanism instills a potential evader with a lack of "certainty about the likelihood of discovery," it can still reduce harm.[3]

This report is written with these goals in mind. Using these mechanisms, actors can better discourage and detect non-compliant activity, increase the cost of spoofing, and build trust in relevant regulations.

Verification Inspection Mechanisms

A common tool in pre-existing arms control verification mechanisms is third-party system inspections to verify compliance with regulation. This first set of mechanisms are intended to support inspections as a tool for AI verification.

Verification Zones and Quarantine Zones

Fundamental to inspections is the ability of third-party inspectors to quickly review a system and target the components that may make it non-compliant. As stated in the introduction, the goal of our verification mechanisms is to demonstrate whether a target system contains AI, and if so, whether certain critical functions of that system can be AI-controlled. This report defines a "critical function" loosely as those functions that regulators believe humans, rather than AI, should control. An example of such a function is the decision over the use of force, which many believe on ethical grounds should be reserved for humans.

Unfortunately, conducting third-party inspections on military and national security systems is often not politically or technically easy and is often dismissed as too difficult to be worthwhile.[4] Several factors complicate matters, including:

- **Complexity**: The high complexity of AI algorithms, and the lengthy code baked into mechanical systems in general, often renders system code difficult to decipher or even unintelligible to humans. Complexity poses a challenge to inspections, a core element of many verification regimes; if inspectors cannot understand the technology they are inspecting, they cannot verify compliance with the regulations they seek to enforce. For AI systems, this complexity is often rooted in the machine learning process, a process by which AI writes and/or adjusts its own algorithms. AI-written code emphasizes functionality, not understanding, creating software not even the most talented engineers, let alone inspectors, can decipher.
- **Invisibility**: AI algorithms run invisibly, leaving no obvious visible markers. Even the physical technologies that underpin AI are opaque. There is no standard set of inputs that "create AI." Each AI system is the unique result of a concert of sensors, communications links, semiconductors, and other technologies. This invisibility makes it difficult to determine whether an AI or a human controls a system or a given function. Further, invisibility poses challenges to the detection of the use and production of AI, complicating efforts to find proof of AI control.
- **Secrecy:** "AI Competition" creates an imperative for states, corporations, and other actors to tightly guard AI algorithms to ensure their advantage.[5] The result has been intense secrecy and a strong disincentive for actors to openly identify which systems are AI systems, let alone expose their inner workings to verification inspectors.

The guiding paradigm of AI verification and inspections has traditionally assumed all systems should be treated and inspected as monoliths. This view is the root of many of the above challenges and has led many experts to contend that AI verification requires full access to a system's software to establish the degree of AI control over it or its functions.[6] Under this assumption, to concretely establish whether a lethal autonomous drone, for instance, allows AI to control the use of force, an inspector would require full access to all software and components contained within. Without full access, the inspector could never be certain if an uninspected part of a system hides an AI program that surreptitiously controls this critical function. Light must be shed on all code to know for certain the system is regulatorily compliant. This view makes inspection an all or nothing task. The noted secrecy demands of national security and proprietary information, however, make full system access politically unlikely. Even if full access is granted, a system-wide inspection would be complex and unwieldy, especially if inspectors must examine long and often indecipherable algorithms. Working under this monolithic systems paradigm, verification does indeed seem impossible.

Thankfully, mechanical systems are not monoliths. The term "system" is no mistake. In reality, any mechanical system is not a monolith but a series of overlapping "subsystems" that usually coordinate under the orchestration of an operating system.[7] Subsystems may include sensors, interfaces, communications, navigation, targeting, and firing, among others. Following this pattern of subdivision, each subsystem often uses its own algorithm(s); some use AI algorithms and others conventional algorithms. As such, a given system can contain no AI, one AI, or many AIs. When tasks are carried out, it is often the subsystems, and their controlling algorithms, that actually implement the overall system's tasks. The operating system merely acts as the system director, allocating resources, easing collaboration, and ensuring smooth operation.

It is unlikely that state actors will consent to inspections of a system's operating system, as it is the centerpiece of any computing system and its design is usually highly secretive, especially in security critical national security systems. The easiest path forward appears to be accepting the operating system as a "black box." That said, inspections still require a window into the overall system, a window that certain subsystems can provide.

The recommendation of this report is to narrow inspection scope to only subsystems of interest to policy goals. This reduces the unwieldy complexity of inspections, balances the need for a certain level of system secrecy, and makes system inspections more palatable overall.

Box 1: Apple iPhone Subsystems

Apple iPhones include both conventional subsystems, such as the internet browser, and AI subsystems, such as Siri. While an iPhone contains an AI, Siri, it is just one application that works in concert with other applications to deliver the various functions and services expected from an iPhone. The iPhone's operating system takes on executive coordination functions, coordinating, triggering, and providing resources for the overlapping subsystems to make sure they work in concert. In sum, the iPhone itself is not an AI.

One can imagine Apple would be much more willing to allow inspectors to view the code for a rudimentary subsystem such as its flashlight application, or any one of its applications for that matter, rather than wholesale access to all of an iPhone's code or its operating system.

To formalize this mechanism, subsystems should be divided into one of two categories, "quarantine zones" and transparent "verification zones," defined as:

Quarantine zones: Subsystems that may contain the operating system, sensitive software, and other system functions intentionally obfuscated due to high secrecy demands. The inner workings of these zones should not matter for verification. Inspectors and policymakers should assume by default that AI exists within or controls these quarantine zones (even if it does not) until proven otherwise.

Verification zones: These are subsystems that are intentionally made transparent. These zones can be physically or logically demarcated within the system through software, hardware, or both. These zones would allow inspectors access to the code or hardware of subsystems to determine from where they receive instructions. If controlling instructions can come from a quarantine zone, it should be assumed the functions in this zone could be AI-controlled. If not, this suggests the subsystem(s) within cannot be controlled by an AI outside of a verification zone. Verification zones should surround subsystems involved in functions that should not be AI controlled or should not contain AI. To verify a human remains in the loop for use of force decisions, for instance, the weapons deployment system should be in a verification zone so an inspector can verify it is not AI controlled and can trace its instructions to a human source.

When curtailing AI command and control, AI itself is often not the problem but rather the ability of AI to communicate with and control critical subsystems. If these critical subsystems are made transparent, inspectors could analyze their code and the source of their commands to determine whether they could come from an AI algorithm. If instructions flow from a quarantine zone, an AI could be sending those instructions, and the subsystem cannot be verifiably AI-free. If the instructions flow only from a transparent source in a verification zone, then inspectors can verify the subsystems within are AI-free or verify what functions are AI controlled within the zone. This mechanism allows for verification that skirts the need to dig through the wholesale code of a system. Through this limited scope, verification by inspection likely grows more politically feasible and practically manageable, while remaining effective.

Verification Checkpoints

Separation of these zones can be implemented either physically or digitally. Physically, zones can be isolated from one another if they do not have the physical means to communicate. This is known as air-gapping.[8] If the weapons system of a drone, for instance, is not wired to a subsystem containing illicit AI, the AI could not control a subsystem or its critical functions.

Digitally, software could be designed to carefully control the flow of instructions and isolate certain software elements from others. I call these verification checkpoints. A verification checkpoint is an interface that stands between a given verification zone and all other subsystems. This checkpoint acts to sieve instructions. Using authentication and carefully managed intra-system communications pipelines, a checkpoint could limit the instructions a verification zone can accept and from where it can accept instructions. If a quarantine zone subsystem is not authorized to communicate with a verification zone subsystem, an instruction checkpoint will halt those instructions. This protects the verification zone from AI within that quarantine zone that may be issuing those instructions. More critically, it demonstrates to an inspector that the AI in the quarantine zone cannot control functions in the verification zone.

Forcing instructions through a checkpoint creates an intentional bottleneck. If all instructions flow through one point, this produces a centralized and convenient node to target all regulation and inspections. A checkpoint's code could be relatively concise and likely technically simple enough for an inspector to easily understand. It is possible that these bottlenecks may negatively impact performance, a tradeoff made in favor of enhanced transparency. Policymakers must consider whether this trade-off is worth it.

As for the specific design of these checkpoints, there is a range of possibilities available. Such checkpoints could be designed to only allow instructions to flow into a verification zone from a select human source. Alternatively, checkpoints could be designed to give varying degrees of control to different sources, perhaps allowing for a benign AI in a quarantine zone, perhaps a navigation AI, to send a verification zone targeting coordinates while continuing to halt the quarantine zone from sending certain critical instructions, such as an instruction to "fire." I call these mixed control checkpoints "centaur checkpoints."

Verification checkpoints reduce potentially unwieldy system inspections to targeted reviews that determine if the checkpoint could allow a quarantine zone, and therefore an AI, to send instructions to a regulated subsystem in a verification zone. By examining checkpoints, inspectors can identify the source of commands to verify if they come from a quarantine zone. If commands can be traced to a human-controlled verification zone, it can be determined that the subsystems beyond the checkpoint are AI-free.

Policy Refinements and Assumptions

Ideally, the quarantine/verification zone would be packaged with some additional policies. If possible, state actors would agree upon standard verification zones in their systems to open to inspectors. State actors must choose and define these zones carefully. To ensure an AI does not have control over critical decisions, states must agree what defines the subsystem that implements those decisions so that it can be clear what is regulated. In general, focus must be placed on subsystems of interest to regulation and those that contain enough information to verify the system will act according to policy goals. To build trust and respect secrecy demands, less is more. Policymakers should work with engineers to determine what quantity of subsystems would need to be verification zones in order to confidently verify a system.

State actors may also wish to negotiate additional policies to ease inspection. Potential examples could include standard subsystem labels and physical markings to clarify component function and layout, designing subsystems to be easily accessed by an inspector, or mandating subsystem code be clearly described in code comments. This is by no means an exhaustive list; state actors should consider these and other policies that may help to ease inspections.

This concept also rests on several assumptions about state actor behavior. It can be assumed that if a state actor knows a given piece of hardware or software is in a verification zone, the state actor will intentionally exclude top-secret technology from that zone's design. Note that opening these zones to inspection does not eliminate the possibility of technical advantage, but merely redirects all actors to finding their technical edge in other non-transparent subsystems. If engineers design systems with this in mind, a state actor should have few reservations about opening these zones to inspection. As addressed below in the "challenges" section, this may mean substantial system changes are required.

Box 2: Inspecting Military Drones through Verification Zones

Say an inspector wants to examine a military drone to verify it cannot use force without a human in the loop. To verify this, actors could agree on select verification zones in this system and open those zones to inspections. To do so, the inspector would require access to only those subsystems relating to the use of force, specifically the weapon deployment subsystem. This subsystem would therefore be a verification zone. Many subsystems, such as the drone's fuel management system, are likely unnecessary to this task and would be declared as uninspectable quarantine zones.

To inspect this verification zone, the inspector may examine the weapons deployment subsystems and determine if it contains AI. Further, the inspector must verify which other subsystems may control weapons deployment and whether those systems are AI-controlled. To do so, the inspector can trace from where inputs into the verification zone flow:

- If instructions are fed to the weapons system from an uninspectable quarantine zone, a zone that could contain AI, the inspector *cannot* verify with certainty that AI does not control the use of force.
- If instructions flow from a transparent verification zone that does not contain AI, then the inspector *can* verify that AI does not control the use of force.

This method allows the inspector to verify compliance while balancing the need for secrecy and reducing the scope of inspection.

Challenges

As mentioned, it is highly likely that these mechanisms cannot be easily implemented using current system designs. Many systems, while modular to an extent, are still highly interconnected, and it may be that certain subsystems cannot be easily subdivided from the rest to form these verification zones. Further, current systems aren't designed with checkpoints in mind, and therefore this could hinder efficiency if instructions must be funneled through this intentional bottleneck. While a hindrance, this problem is not something that should halt research and policy negotiation. Historically, it has taken many parties to arms control treaties significant time and research to implement the changes required. For example, the United States is still working through the substantial changes required by the Chemical Weapons Convention it signed in 1997 with a goal of completing disarmament by 2023.[9] This demonstrates that just because arms control verification requires heavy lifting and time does not render it impossible, nor does it mean steps should not be taken. Verification and arms control problems are technical and require technical changes. Therefore, it is likely these changes must be gradually implemented over time as new national security and military systems are developed.

Cybersecurity is another challenge. With greater transparency comes greater cyber risk. When considering this mechanism, state actors must consider the security of their systems and the benefits they may receive by sacrificing some system security. Arms control usually has some cost, and it could be that cost is cybersecurity. Policymakers must note, however, that often security in one domain can be improved through a small sacrifice of security in another. Weigh this balance carefully.

A final challenge is one of state actor trust and motivation. Any arms control scheme that seeks to balance secrecy and transparency will encounter international resistance. Change is hard, especially when it costs time, money, and potential military advantage. States must be motivated to buy into the concept and must trust the technical decisions made throughout the process. Without motivation and trust, negotiations will fail.

To surmount these challenges, policymakers should work with engineers to research which checkpoint designs and system architectures will best balance policy goals against any detriments to cybersecurity and processing speeds. To improve trust and build international buy-in, policymakers should collaborate on this research with international partners. Doing so will ensure designs reflect the state of the art, build trust in the science behind any proposed changes, and encourage buy-in from the outset of the development process.

Hardware Inspections

If state actors wish to avoid software inspections and verify based only on hardware, they could do so by examining whether a given system, or subsystem, contains an AI chip. This mechanism can be implemented using the verification/quarantine zone model, restricting inspectors to only inspecting the hardware of subsystems relevant to their purpose. If a given subsystem is not powered locally by an AI chip and verifiably cordoned off from any external AI, it can be assumed the subsystem and its functions are not AI controlled. If the goal is to simply determine whether a system writ-large is AI-free, this mechanism is ideal as its somewhat less intrusive nature than software inspections[*] may make it more politically feasible to inspect an entire system without risking the exposure of most of the system's design secrets.

The mix of chips that inspectors would look for in systems is diverse, ranging from general purpose Central Processing Units (CPUs) to so-called AI chips, which include Graphics Processing Units (GPUs), highly specialized Application-Specific Integrated Circuits (ASICs), and other AI optimizing processors and memory units.[10] In the world of AI, these various chips fall into two buckets: those used for AI training, when the AI is in development, and those used during inference, when the AI is in use. For AI system verification purposes, it is the inference chips that matter most as those are the chips that power AI algorithms when AI systems are deployed.

At present and into at least the short term, it seems likely these chips will be required for AI system inference and therefore a clear indicator of the use of AI by that system. McKinsey estimates that within five years, 70 percent of the chips in AI systems will be ASICs, a specific type of AI chip exclusively used by AI applications, replacing the CPU as the AI processor of choice.[11] The takeaway is that within five years, most AI embedded systems could clearly be identified by their chipset.

The reason for this shift, in brief, is that AI chips appear to offer many clear advantages. The use of these chips for AI processing and specifically AI

[*] It is assumed that because the complexity and sensitive design details of hardware is often invisible to the naked eye an inspector could not easily steal or reveal substantial design secrets unless they had the time and resources to analyze the components in depth. Inspecting software, on the other hand, requires an inspector to read through the system's code, revealing all the design secrets contained within. It is likely certain secrecy risks, however, are not accounted for in this report. Policymakers should work with engineers to verify what secrets may be put at risk through a hardware inspection.

inference largely derives from the special purpose features they provide for AI systems. These include (but are not limited to):

1. **Greater parallelism.** To analyze data efficiently, AI algorithms often must take advantage of greater parallelism than traditional CPUs can offer. These algorithms need to subdivide their complex input data and analytic models into smaller pieces so that analysis can be done in parallel, rather than sequentially. This massively speeds up operations and allows for quicker analysis.
2. **Lower precision numbers:** The operations AI algorithms perform often require fewer values after the decimal point in the numbers being calculated. These numbers are said to have "lower precision." If a chip lowers the precision of the numbers being calculated, this increases the speed of each calculation, unlocking improved performance.[12]
3. **Faster memory access:** AI has significant memory requirements. Fast memory allows an AI to think and act faster. AI chips facilitate this, often physically shortening the distance between processing and memory components to increase the speed of memory recall and, therefore, the speed of AI "thinking."[13]
4. **Optimization for AI specific calculations:** AI computations often involve a high volume of matrix multiplication. AI chips often optimize for this through such features as many parallel Multiply-Accumulate Circuits, which are special circuits that speed up this type of multiplication.[14]

These features illustrate how AI chips can improve function in AI applications and why they may become an essential part of national security AI systems.

As alluded to, inspections to verify the existence of AI in a system could be simple. All that is required is a physical examination of a system or subsystem to check if it contains AI chips. Verifying AI control over specific critical functions, such as an AI's ability to use force, is also potentially possible from this "look and see" method. If a critical subsystem, such as the weapons deployment system, does not include an AI chip and is verifiably cordoned off or air-gapped from other subsystems, it is likely not AI controlled and therefore compliant.

While potentially effective, this mechanism is blunt. It is likely best used as a "first pass" inspection to determine whether a system requires greater scrutiny. Additional, more invasive analysis could then follow.

Challenges

This mechanism does not *guarantee* systems are entirely AI-free, as traditional CPUs can still run AI algorithms, albeit at an often slower pace. Furthermore, the utility of this mechanism could vary widely depending on system type. For lightweight applications, such as small drones, CPUs may be enough, and policymakers should account for this in their inspection policies. In this case, seeing that a system does not contain an AI chip is not enough for verification.

However, for heavyweight applications, such as large drones and autonomous vehicles, the real time requirements of safety and time-critical functions require immensely fast processing to achieve real time action and reaction. This may require the faster AI-specific inference chips.[15] This speed imperative is even more crucial to military systems where the inability to react in a timely manner could lead to strategic failure or dangerous lag. It can be assumed that if a military deploys a multimillion-dollar autonomous system, it will use the chipset best equipped to reap the value of that investment. If current trends hold, this means those systems will use AI chips. In the end, capability is what matters. If a CPU-driven autonomous system is not capable of the processing speeds needed to be a true threat, it may be of little concern to regulators and useless to its owners. Do note that current trends may change. Refinements in CPUs and efficiency gains from new AI design techniques could allow systems to run on CPUs. If this mechanism is used, policymakers must ensure implementation keeps pace with technology change.

Another challenge is distinguishability. This mechanism depends on AI chips exhibiting some quality that allows an inspector to determine they are AI chips. Given the breadth of AI-specific chips, this will likely require a range of qualities that inspectors can use to determine the classification of a given chip. As such, policymakers should work with engineers to delineate what qualities can be used and ensure this process conforms to chip designs.

Sustained Verification

A key challenge to verifying systems remain compliant with regulations post inspection is the ease with which system code can be altered. The inspection mechanisms above are certainly essential, especially for deterring non-complaint activity. However, inspections only offer certainty in the very short term. Even if a system is found to be compliant, its software could be changed immediately after to embed it with AI or give an AI control of critical functions. Effective inspections and regulatory enforcement, therefore, require mechanisms to detect and deter such alterations. To craft a strong verification regime, policymakers should look to what I call "sustained verification mechanisms," which help affirm previously inspected systems remain compliant long term.

The following mechanisms are tailored to this purpose.

Cryptographic Hashing

One mechanism that can be used to protect system code from manipulation is anti-tamper (AT) software techniques. According to the Department of Defense, these methods render efforts to alter or exploit data and code "so time consuming, expensive, and difficult that even if the attempts were to become successful, the AT protected technology will have been updated and replaced by the next generation version."[16] For AI verification, anti-tamper software ideally can even "detect if [a] program has been altered," and potentially rendered non-compliant.[17] These methods promise to deter and detect rather than outright stop tampering. Therefore, policymakers should note that software tampering is still possible, despite the best efforts of anti-tamper technology.

Anti-tamper methods are diverse, often application-specific, and continually evolving. As such, this report does not address many existing methods, such as watermarking, which makes code difficult to remove without damaging the system or making changes, or encryption wrappers, which encrypt system code and only decrypt it when the system is in use.[18] For the sake of illustration, I discuss two promising anti-tamper techniques—cryptographic hashing in this section and algorithm obfuscation in the next—and how they can be used for sustained AI verification. Interested policymakers and researchers are encouraged to research and consider other AT methods that could be appropriated for verification.

Cryptographic hashing is a tool that could support verification by producing for inspectors a privacy-preserving "record" of system code. This record can act as a tamper detection device, allowing tampering to be signaled upon subsequent inspections of the same system.

Cryptographic hashing uses a mathematical function, known as a "cryptographic hash function," which takes in any type of data—including text, audio, and computer code—then scrambles, condenses, and transforms that data into a resulting "hash," a seemingly random sequence of characters of a certain length.[19] For example, when the 39,808-word text of the Book of Genesis, King James Version, is fed through the SHA-256 hash function (a commonly used hash function), the entire text is transformed into: "675d773189394dcd4cc92d1b489f1e04cca2b4e734dccda7d7e06d0aed181db8."[20] This resulting hash is an unintelligible scramble, dramatically condensed and altered from the text.

This data transformation property is only the tip of the cryptographic hashing iceberg. Importantly, for a hash function to be a *cryptographic* hash function and useful for verification, it must have five key characteristics:

1. **Consistency**: Input data will always produce the same hash when run through the same cryptographic hash function.
2. **Speed**: The hash function is fast, producing output in a reasonable amount of time.
3. **Cannot be reverse-engineered:** The resulting hash cannot be easily transformed back into the original input data that it represents.
4. **Cannot be easily duplicated:** Practically speaking, no two data inputs should produce the same hash.
5. **The Avalanche Effect:** Only a small alteration to the original input data will yield a dramatic "avalanche" of changes to the resulting hash. Small changes yield substantial differences.

For sustained verification purposes these combined properties create a very powerful tool. Once an inspection is completed, an inspector can hash the entire code of a system, or even just the code of a verification zone, and store that hash as a "record" of what that code should be. To an inspector, the hash *is* the code. If inspectors have records of expected system code, this then allows them to identify and track system changes. If the same system's code were to be hashed again in the future, the result should match the hash the inspector has on file. If it does not, this signals alterations have been made. Crucially, this tool is privacy preserving. Because a given hash cannot be reverse-engineered to produce the code it represents, an inspector can save a record without putting state secrets at risk.

It is notable that hashes do not need to be computed in person. This opens up the opportunity to use this tool not only for sustained verification, but *continuous verification*. To implement, a given system could be fitted with a theoretical "hashing device" that occasionally hashes the system's code and broadcasts that hash to an observer. This would give the observer a window into the code's fidelity over time and deter actors from making changes. Naturally, this concept would be somewhat invasive but would create a powerful verification tool.

Challenges

It is possible that, over time, certain cryptographic hashing functions may be broken, thereby compromising the integrity of this system. In the past, even hashing algorithms supported by reputable organizations such as the National Institute for Standards and Technology have contained compromising vulnerabilities.[21] If a significant vulnerability is found, code could theoretically be altered without signaling tampering. Therefore, an actor could alter their system code in such a way that it produces the same hash as the original and evade detection. This could facilitate cheating.

This scenario, however, is immensely improbable. The mathematics of the best hashing algorithms are such that even if an actor were to try and fool the function, the process would take so long (perhaps decades) that the effort is not worthwhile.[22] The improbability of cracking existing top-tier hash functions is illustrated best by Bitcoin, the mechanics of which are built on cryptographic hashing. There exists an unparalleled monetary incentive to crack Bitcoin's hash function, yet the combined efforts of the world's hackers have failed thus far. Hashing is that robust. Still, if this method were implemented, policymakers should consider the probability of vulnerabilities and prepare appropriate contingencies.

Algorithm Obfuscation

The second anti-tamper technique I will discuss for the sake of illustration is algorithm obfuscation.

Algorithm obfuscation is a method that "make[s] a computer program 'unintelligible' while preserving its functionality."[23] Obfuscation is achieved by running code through an obfuscation algorithm, which intentionally scrambles the code to disguise its meaning. The resulting scrambled code is an unintelligible “black box,” which can function exactly like the original code but cannot be read by engineers and, ideally, cannot be reverse-engineered to discover the unscrambled version. The difference between obfuscation and hashing is that hashing would produce a numeric representation of the code (which cannot be computed), while obfuscation produces new, unintelligible code (which can be computed) with the exact functions of the original code.

For verification, obfuscated code would act in a similar manner to cryptographically hashed code. If code is truly obfuscated, outside parties and inspectors could potentially view a system’s code, and even record it, without compromising its specific implementation. Inspectors could keep a record of the entire system code, or just the code of a verification zone, and therefore a record of how that system should be configured. In subsequent inspections, inspectors could run a 1-to-1 comparison between past and present obfuscated code to quickly spot tampering. Unlike hashing, however, obfuscation allows inspectors to see code, not a hash, potentially eliminating the problem of cryptographic hash collisions.

Challenges

Code obfuscation is still a developing cybersecurity frontier, and much research is needed to produce robust obfuscation algorithms. According to the Defense Advanced Research Projects Agency (DARPA), the best current techniques only require about a day’s worth of effort to crack.[24] That said, it is possible that adequate technology has been or will be developed. DARPA is currently developing what it calls “Safeware,” obfuscated software that cannot be reverse-engineered and is probably secure.[25] While this project is a moonshot, if it, or any other related research, is successful and made internationally available, state actors could use algorithm obfuscation to ensure these systems are secure, tamper-proof, and help build trust in regulations.

Even if robust obfuscation techniques are indeed developed, it is important to note that this method does reveal code and, with it, certain elements of code structure and function. Further, because this is usable code, it could be used to determine what functions a given system has, even if it cannot be determined how the system performs those functions. Additionally, one could still find vulnerabilities in obfuscated code even if the overall purpose of a given algorithm remains unclear. All in all, in its current state obfuscated code is likely a less secure method than cryptographic hashing. However, it still represents one potential tool in the anti-tamper toolbox that can be further refined through consultation with engineers.

Van Eck Radiation Analysis

Van Eck radiation analysis is another sustained verification tool that can be used to continuously monitor system function and ensure it matches what was found during a system inspection. Van Eck radiation is the electromagnetic radiation computers and unshielded electronic devices emit when they process code. If intercepted, this radiation can be deconstructed to reveal a system's functions and even the code it processes.[26] Such potent information can be used to verify the consistency of the system's code in the long term, creating a powerful verification tool.

As with most verification tools, however, Van Eck radiation is not a silver bullet. While computer code can indeed be intercepted by analyzing Van Eck radiation, this code will be highly complex and often garbled. Computers are excellent multi-taskers and often process multiple algorithms in tandem, making it difficult to piece together coherent algorithms from a tangle of intercepted instructions. This quality rules out any ability to simply identify algorithms using Van Eck radiation. For this, Van Eck radiation analysis is far too blunt an instrument.

To further complicate matters, the instructions Van Eck radiation would reveal are written in machine code, the strings of 1s and 0s that form the operating instructions for computers. Machine code instructions are often processor-specific, meaning that if an analyst is unfamiliar with the processor running this machine code, they will not know how to correctly interpret the intercepted 1s and 0s. In sum, if the Van Eck radiation of a complex system such as a military drone were monitored, it is unlikely an observer could piece together what algorithms the system is running, let alone prove whether it contains AI or gives AI control over certain functions.

At first glance, these two qualities may seem to rule out Van Eck radiation analysis as an effective verification tool. In reality, however, it is precisely Van Eck monitoring's weaknesses that make this highly practical for AI verification. Van Eck radiation leaks *just enough* information about system code, without revealing the complex, tightly guarded secrets of the system's algorithms.

To use Van Eck radiation to sustain verification post-inspection, a system's owner would need to consent to the installation of a theoretical "Van Eck radiation sensor" onto the system in such a way that it can detect the unshielded Van Eck emissions of the device. This sensor would need to be trained to recognize the typical radiation patterns given off by this system, creating a baseline system radiation profile. This profile would consist of a "dictionary" of expected radiation patterns emitted when the system uses its existing code. Each time the sensor picks up a Van Eck emission, it would consult this dictionary to affirm that the emission matches the patterns expected of the system. If it detects a foreign pattern, be it from injected malware or intentional design changes by its owner, this indicates changes have been made to system code. The Van Eck radiation would then signal to inspectors that the system may no longer be compliant.

To illustrate, if an inspector verifies a drone as AI-free, the inspector could then use a Van Eck monitoring device to measure the radiation patterns the drone's code emits when its algorithms are run. If the drone's radiation patterns later change, this would be detected. Inspectors could then re-inspect the drone, determine the cause of the deviation, and establish whether the system remains compliant.

The unique advantage of Van Eck radiation analysis over other sustained verification tools is its potential for minimally invasive monitoring. A theoretical Van Eck sensor would not need access to a system's code to verify that code has been altered. Detecting system changes needn't come at the expense of code secrecy. System monitoring can be done mostly externally (assuming the sensor can be installed without a radiation shield in its way) without reading a single line of code. This protects design secrets while guarding against illicit activity.

This mechanism is scientifically grounded. Recent DARPA-funded research tested a nearly identical concept, albeit with the goal of detecting malware rather than changes to AI control of a system. The research found that illicitly injected malware could be detected in systems more than 99 percent of the time.[27] If this same process is appropriated for AI detection, systems could be continuously monitored and non-compliant activity detected with certainty beyond a reasonable doubt.

Challenges

Van Eck radiation's use in intelligence collection is noted for its difficulty. Challenges certainly remain, many of which must be tackled by engineers to determine the effectiveness of this method and the quality of information that can be collected. One such challenge is the known changes to emissions that result from integrated circuit wear and tear over time.[28] Such changes would need to be studied, quantified, and their impact accounted for in any Van Eck radiation analysis system. Another challenge is the potential impact of different environmental conditions on Van Eck emissions. These may include the impact of power lines, radar jamming equipment, and other stray electromagnetic fields. These influences would need to be accounted for, otherwise a system could signal changes every time it enters a foreign operating environment.

The applicability of this measurement specifically to machine learning and deep learning models must also be studied in depth. One such challenge may be accounting for emissions changes resulting from the dynamic alterations a deep learning model may make on itself. Another could be accounting for a more diverse range of emissions created by the varied data these systems process.

As mentioned, AI verification is a technical problem that needs technical solutions. Van Eck radiation analysis is a method that must be developed in consultation with engineers and specialists to ensure it provides robust results and accounts for a variety of implementation scenarios and challenges. Such challenges to verification development are not unheard of. Thankfully, in verification, half measures are acceptable. This mechanism does not need the precise accuracy to concretely determine whether a system was manipulated to be an effective tool. If it can detect potential manipulation with a certain level of uncertainty, this can still signal to observers that follow-up action may be required.

Conclusion

AI verification is no easy task. The mechanisms discussed in this report offer potential solutions to some of the many problems that face AI systems. More work is required, however. If the goal of policymakers is to regulate AI, there are actions that can be taken today. Specifically:

1. **Verification Zone and Hardware Verification Architecture Research:** Experts must develop and research system architecture that can be used to implement the verification zone and hardware verification concepts. Research should specifically identify what architectural options exist to cordon off and checkpoint the verification zones, the impact of these checkpoints on system performance and security, the information that can be gained or lost using these inspection mechanisms, and the effort it would take to implement these architectures in future and current state systems.
2. **Van Eck Radiation Capture and Analysis Research:** Further research is needed to determine the quality of information that can be captured from the Van Eck radiation given off by systems during processing and the degree of certainty of system consistency that information provides. Additional research should focus on the feasibility of a "Van Eck radiation sensor" to be used for continuous verification.
3. **Coordination with International Partners**: To build trust, researching, developing, and implementing these concepts cannot be done in a vacuum. Arms control requires international support, and trust requires transparency in the science. The United States should explore potential research partners to build support in these ideas, verify research, and foster trust in potential arms control agreements that may follow.

By taking these steps and working with scientists, policymakers can move the AI arms control conversation forward. AI is at a fulcrum point, and international security depends on ensuring AI provides a net benefit. Policymakers must take it upon themselves to explore these ideas and build on this research so that robust and verifiable standards, norms, and regulations can be developed to constrain the misuse of AI systems and preserve future security.

Author

Matthew Mittelsteadt is an Artificial Intelligence Policy Fellow for the Institute for Security Policy and Law (SPL) and a guest lecturer for the Syracuse University College of Law.

Acknowledgments

I want to extend special thanks to James E. Baker, Laurie N. Hobart, Igor Mikolic-Torreira, Eric Landree, Matthew Mahoney, Dewey Murdick, Alex Friedland, and Miles Brundage for their invaluable input, numerous read-throughs, and time. This wouldn't have been possible without your effort. An extra special thanks to my ever-inspiring grandma, Charlotte Gronseth, who proved that even an octogenarian can understand and offer useful critique on artificial intelligence research.

CSET Product ID #: 20190020

Document Identifier: doi: 10.51593/20190020

Endnotes

[1] Ola Dahlman, "Verification: To Detect, to Deter and to Build Confidence," United Nations Institute for Disarmament Research, *Arms Control Verification* (Geneva: UN, October 2010), 3-13, https://www.unidir.org/files/publications/pdfs/arms-control-verification-en-320.pdf.

[2] "The Chemical Weapons Convention: Report Together with Majority and Minority Views (to Accompany Treaty Doc. 103-21)." United States Congress, Senate Committee on Foreign Relations: U.S. Government Printing Office, 1996.

[3] United Nations Institute for Disarmament Research, *Arms Control Verification*.

[4] Mark Gubrud, "Can an Autonomous Weapons Ban Be Verified?" International Committee for Robot Arms Control, April 14, 2014, https://www.icrac.net/can-an-autonomous-weapons-ban-be-verified/.

[5] Andrew Imbrie, James Dunham, Rebecca Gelles, and Catherine Aiken, "Mainframes: A Provisional Analysis of Rhetorical Frames in AI" (Center for Security and Emerging Technology, August 2020), https://cset.georgetown.edu/wp-content/uploads/CSET-Mainframes-A-Provisional-Analysis-of-Rhetorical-Frames-in-AI.pdf.

[6] Gubrud, "Can an Autonomous Weapons Ban Be Verified?"

[7] Terry Fong, "Autonomous Systems: NASA Capability Overview" (National Aeronautics and Space Administration, August 24, 2018), https://www.nasa.gov/sites/default/files/atoms/files/nac_tie_aug2018_tfong_tagged.pdf.

[8] Kim Zetter, "Hacker Lexicon: What Is an Air Gap?" *WIRED*, December 8, 2014, https://www.wired.com/2014/12/hacker-lexicon-air-gap/.

[9] "Chemical and Biological Weapons Status at a Glance," Arms Control Association, April 2020, https://www.armscontrol.org/factsheets/cbwprolif.

[10] Saif M. Khan and Alexander Mann, "AI Chips: What They Are and Why They Matter" (Center for Security and Emerging Technology, April 2020), https://cset.georgetown.edu/research/ai-chips-what-they-are-and-why-they-matter/.

[11] Gaurav Batra, Zach Jacobson, Siddarth Madhav, Andrea Queirolo, and Nick Santhanam, "Artificial-Intelligence Hardware: New Opportunities for Semiconductor Companies," McKinsey and Company, January 2, 2019, https://www.mckinsey.com/industries/semiconductors/our-insights/artificial-intelligence-hardware-new-opportunities-for-semiconductor-companies#.

[12] J. Welser, J.W. Pitera, and C. Goldberg, "Future Computing Hardware for AI," 2018 IEEE International Electron Devices Meeting (IEDM), December 1-5, 2018, https://doi.org/10.1109/IEDM.2018.8614482.

[13] Anirudh VK, "Why ASICs Are Becoming So Widely Popular for AI," *Analytics India Magazine*, July 15, 2019, https://analyticsindiamag.com/why-asics-are-becoming-so-widely-popular-for-ai/.

[14] Khan and Mann, "AI Chips: What They Are and Why They Matter."

[15] Khan and Mann, "AI Chips: What They Are and Why They Matter."

[16] Christopher L. Cain, "Anti-Tamper Technology: Preventing and/or Delaying Exploitation of Critical Technologies," Utica College, 2013, https://www.utica.edu/academic/library/Cain_CL_2013.pdf.

[17] C.S. Collberg and C. Thomborson, "Watermarking, Tamper-Proofing, and Obfuscation – Tools for Software Protection," *IEEE Transactions on Software Engineering* 28, no. 8 (November 2002): 735-746, https://ieeexplore.ieee.org/document/1027797.
[18] Mikhail J. Atallah, Eric D. Bryant, and Martin R. Stytz, "A Survey of Anti-Tamper Technologies," *CrossTalk: The Journal of Defense Software Engineering* (November 2004): 12-16,
http://static1.1.sqspcdn.com/static/f/702523/9292150/1289015004337/200411-Atallah.pdf?token=n6JBO1DGr6SY5KwDbkykN4ih4wg=.
[19] Quynh Dang, *Recommendation for Applications Using Approved Hash Algorithms* (Washington, DC: Department of Commerce, August 2012),
https://www.nist.gov/publications/recommendation-applications-using-approved-hash-algorithms?pub_id=911479.
[20] *Genesis, King James Version*, accessed December 18, 2020,
https://quod.lib.umich.edu/cgi/k/kjv/kjv-idx?type=DIV1&byte=1477; "SHA2 Hash Generator," Browserling, accessed July 17, 2020,
https://www.browserling.com/tools/sha2-hash.
[21] Information Technology Laboratory, *Secure Hash Standard (SHS)* (Gaithersburg, MD: National Institute of Standards and Technology, August 2015), FIPS PUB 180-4,
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf; Liberty York, "What is Hashing?" *Medium* (blog), February 22, 2018, https://medium.com/tech-tales/what-is-hashing-6edba0ebfa67.
[22] Nathan Landman, Christopher Williams, Eli Ross, and Jimin Khim "Secure Hash Algorithms," Brilliant, accessed February 4, 2020, https://brilliant.org/wiki/secure-hashing-algorithms/.
[23] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters, "Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits," International Association for Cryptologic Research (IACR), July 21, 2013,
https://eprint.iacr.org/2013/451.pdf.
[24] Joshua Baron, "SAFEWARE," Defense Advanced Research Projects Agency (DARPA), accessed December 23, 2019, https://www.darpa.mil/program/safeware.
[25] Baron, "SAFEWARE."
[26] National Security Agency, "TEMPEST: A Signal Problem," accessed December 23, 2019,
https://www.nsa.gov/Portals/70/documents/news-features/declassified-documents/cryptologic-spectrum/tempest.pdf.
[27] Haider Adnan Khan, Nader Sehatbakhsh, Luong Ngoc Nguyen, Robert Locke Callan, Arie Yeredor, Milos Prvulovic, and Alenka Zajic, "IDEA: Intrusion Detection through Electromagnetic-Signal Analysis for Critical Embedded and Cyber-Physical Systems," *IEEE Transactions on Dependable and Secure Computing* (August 2019): 1–1,
https://doi.org/10.1109/TDSC.2019.2932736.
[28] Alexandre Boyer, Sonia Ben Dhia, Binhong Li, Néstor Berbel, and Raul Fernandez-Garcia, "Experimental Investigations into the Effects of Electrical Stress on Electromagnetic Emission from Integrated Circuits," *IEEE Transactions on Electromagnetic Compatibility* 56, no. 1 (February 2014): 44-50, https://ieeexplore.ieee.org/document/6563162.