Interim Report: Mechanisms for Flexible Hardware-Enabled Guarantees

James Petrie, Onni Aarne[1], Nora Ammann[2], David 'davidad' Dalrymple[2]

August 23rd, 2024

[1] Institute for AI Policy and Strategy
[2] Advanced Research and Invention Agency

Executive Summary

As general purpose AI technology develops, it may come to have far-reaching implications for international security, necessitating international governance. International governance of technology is generally difficult, and AI may, by default, be particularly difficult to govern due to its intangible and general-purpose nature.

We propose Flexible Hardware-Enabled Guarantee (flexHEG) mechanisms, which could be added to AI accelerators to enable multilateral, privacy-preserving and trustworthy verification and automated compliance guarantees for agreements regarding the development and use of advanced AI technology. To ensure that all parties could trust these mechanisms, they would be fully open source and auditable, while also being robust to tampering even from state-level adversaries.

This trustworthy and secure design means flexHEG-capable *guaranteeable chips* could enable genuinely multilateral control over frontier AI technology. Mutually agreed-upon rules could be set and updated through a multilateral and cryptographically secure mechanism in order to guarantee that only agreed-upon restrictions are deployed. The flexible, general-purpose nature of the mechanism would allow rules to adapt to future developments.

Concrete policy options flexHEG could enable include:

- Limiting the size of training runs in terms of total FLOP
- Limiting the size of datasets that can be used in a training run
- Privacy-preserving verification that certain types of training data are not being used
- Privacy-preserving verification that certain model architectures or training methods are or are not used
- Requiring the possession of a non-expired license to run computations of a certain size range
- Requiring a standardized evals protocol to be incorporated into the computation graph of training for sufficiently large training runs
- Controlling access to model weights by requiring exported model weights to be encrypted

Several technical problems would need to be solved to create sufficiently effective and secure flexHEG mechanisms. We are confident that some variant of this proposal is technically feasible, however, further research is needed to determine whether it can be made sufficiently secure without excessive impact on the price-performance of the overall package. In order to ensure that the technology is available by the time it is needed, R&D efforts to develop production-ready flexHEGs would ideally begin as soon as possible.

Table of Contents

Introduction

"Unsafe development, deployment, or use of AI systems may pose catastrophic or even existential risks to humanity within our lifetimes. These risks from misuse and loss of control could increase greatly as digital intelligence approaches or even surpasses human intelligence.

In the depths of the Cold War, international scientific and governmental coordination helped avert thermonuclear catastrophe. Humanity again needs to coordinate to avert a catastrophe that could arise from unprecedented technology."

— International Dialogues on AI Safety, joint statement, Beijing, March 2024

Future AI systems could pose serious risks to public safety and international security. This has increasingly given rise to calls for domestic regulation and international governance of AI.

International governance could create trust that all relevant parties are taking appropriate precautions when developing and deploying advanced AI. However, there are corresponding concerns about the feasibility of such governance. Agreements in international contexts are famously challenging due to the lack of trusted third parties to help verify and enforce compliance.

AI accelerators are a natural node for AI governance because their supply chain is extremely concentrated and huge quantities are needed to train frontier AI models [1]. Governance could be supported by mechanisms built directly into AI accelerators so that restrictions on unsafe usage are much harder to circumvent [2], [3].

We propose **Flexible Hardware-Enabled Guarantee (flexHEG) mechanisms**, which could be added to AI accelerators to enable multilateral, privacy-preserving and trustworthy verification and automated compliance guarantees for agreements regarding the development and use of advanced AI technology, thereby substituting for the missing trusted third party.

This trustworthy and secure design means flexHEG-capable *guaranteeable chips* could enable genuinely multilateral control over AI technology, thus making it possible for a range of stakeholders to agree on a variety of potential rules, from safety rules to robust benefit-sharing agreements. Mutually agreed-upon rules could be set and updated through a multilateral and cryptographically secure mechanism in order to guarantee that only agreed-upon rules are applied. Guaranteeable chips would also enable various parties to make specific cryptographically verifiable claims that could prove compliance with agreements.

Examples of regulatory capabilities that guaranteeable chips could enable include:

- Limiting the size of training runs in terms of total FLOP
- Limiting the size of datasets that can be used in a training run

- Privacy-preserving verification that certain types of training data are not being used
- Privacy-preserving verification that certain model architectures or training methods are or are not used
- Requiring the possession of a non-expired license to run computations of a certain size range
- Requiring a standardized evals protocol to be incorporated into the computation graph of training for sufficiently large training runs
- Requiring model weights to be encrypted in such a way that only allows them to be used on (specific) other flexHEG-capable devices, optionally according to specific rules, thereby enabling improved security and effective governance of even distributed AI training & inference

A perennial problem in the governance of technology is that it is difficult to predict what kinds of rules and verification might eventually be needed as the technology develops. The programmable nature of flexHEGs would enable the implementation of a wide range of agreed-upon rules, thereby allowing governance to adapt to future developments, and extending the range of feasible agreements. This flexibility could also help to avoid the accruing of outdated, overly blunt or ineffective regulation.

Mechanisms for flexHEGs could also enable qualitatively improved safety and security by implementing rules locally on-chip and thus allowing rule violations to be prevented from occurring in the first place. In addition to reducing the need to rely on intrusive monitoring and costly after-the-fact punishment to deter violations, local implementation of rules could be particularly valuable if AI technology becomes so powerful that a rule violation could have consequences — intentional or not — that would be so far-reaching that it could not be recovered from, or would undermine the ability for any actor to punish it.

Mechanisms for hardware-enabled guarantees should only be implemented in a form that would address concerns about privacy, security, and risks of regulatory overreach. We discuss these issues at more length in Appendix B.

To ensure that all parties could trust these mechanisms, the flexHEG design and guarantee mechanisms should be made fully open source so that their integrity can be validated via external audits. Additionally, the guaranteeable chips should be robust to tampering even from state-level adversaries, to ensure that even state actors cannot compromise the mechanisms. This would allow all parties to establish justified trust that the mechanism only does what it is expected to do, and that it cannot be used as a "backdoor" to spy or intervene on chips arbitrarily or without the owner's knowledge.

To ensure privacy and security, the flexHEG design would never involve "phoning home" without permission and review by the device owner. This means that no information can be secretly collected about the accelerator without the device operator being aware. Local implementation of rules (as discussed above) also greatly reduces the need for any information to be reported to regulators or other third parties. Nearly the only information that would need to go from a regulator to the secure processor would be firmware updates, which could be compiled from open source code by the chip owner. The only exceptions to this would be some specific information possibly required to implement specific guarantees, but this information could also be controlled and reviewed by the device owner. Overall, it should be feasible to operate guaranteeable chips even in an air gapped environment.

To ensure that all devices implement the latest rules, it could be desirable for the mechanism to require regular updates to continue operating. However, this would raise concerns about users being forced to install arbitrarily restrictive rule sets in the future. This is an important problem, but could likely be addressed in various ways, such as one or more of the following:

- The mechanisms could be configured to only accept updates that have been signed by a quorum of parties. These parties could be parties in an international process designed to ensure that rules are only implemented if they are genuinely necessary for international security.
- The mechanisms could be configured to irrevocably accept a roll-back to some minimal baseline rule set such as one that would restrict the total size of clusters, and nothing else. We discuss how a maximum cluster size rule could be implemented in Appendix A.

The practicality of flexHEG mechanisms depends on non-trivial engineering R&D. A major focus of this report is on the different areas of technical work that are needed to build guaranteeable chips. Based on this analysis, we believe that guaranteeable chips will be feasible to implement, although at the time of this writing we are still uncertain about the level of security and verifiability that will be achievable (which will dictate which contexts they are suitable to be used in).

This research and development work will likely take several years. For this reason, we believe that development efforts should begin as soon as possible. While current AI technology is not yet powerful enough to require strong forms of international governance, two years ago there was no apparent need for an international AI Safety Summit or national AI Safety Institutes, and we may find that in a few years' time the AI field has progressed enough to make international governance desirable.

In the meantime, the technological advancements this research program would produce will likely be useful for improving security and trust in both commercial and government applications. The flexHEG design can be seen as an improved and extended form of confidential computing, which has already been used by the computing industry to improve security and enable privacy-preserving collaboration.

About This Report

The primary goals of this interim report are to explain the flexHEG design proposal and to assess its overall feasibility.

Having discussed the central motivations for guaranteeable chips in the preceding introduction, we will next provide a high-level conceptual overview of the full flexHEG design stack and discuss the types of policy options this would enable. We will then provide an overview of the key technical objectives which a consolidated R&D effort to develop the full flexHEG capabilities would have to address. Next, we will present a concrete architecture proposal that we believe to be particularly feasible, and identify directions for future research. We also include two appendices: Appendix A discusses in more detail what guarantees the flexHEG design would enable, and Appendix B further discusses concerns about privacy, misuse and security.

Because flexHEG mechanisms allows the deferral (and later revision) of what specific rules or policies should be implemented using these mechanisms, this report will not discuss these questions in detail. We do provide outlines of some policies that could be implemented using these mechanisms in order to illustrate the mechanisms' capabilities.

Alongside this report, there is an ongoing effort to build and test a proof-of-concept of the proposed architecture, using off-the-shelf components, with a fully open source hardware and software stack, for less than $5,000 per unit (with the majority of that cost being the AI processor). More information on this can be found at this website. The prototype is not intended to meet all the security requirements of our proposal in full, but to provide it as a starting point for iteration by other researchers, and for more grounded conversations with interest groups in the field.

Conceptual Overview of the FlexHEG Design Stack

In this section, we will give an initial overview of the full flexHEG design stack. We will focus on clarifying the key components and their respective functions before, in later sections, discussing how the different components could concretely be implemented in more detail.

In the flexHEG design, each of the AI accelerator chips (such as GPUs) would be placed in a tamper-proof enclosure along with an auxiliary *secure processor*. All of the accelerator's memory and other components would also be contained within the enclosure.

The secure processor would be an open-source, standardized, general-purpose processor that sits between the accelerator chip and the rest of the world, able to locally access all information and instructions going to and from the chip as well as some aspects of the accelerator's state. By selectively blocking and encrypting incoming and outgoing information as needed, the processor could implement a wide range of guarantees.

Crucially, the secure processor would be able to encrypt and authenticate data coming from the accelerator chip and from the secure processor itself, and thus enable privacy-preserving, sophisticated, and programmable verification schemes. Encryption would allow improved security through preventing anyone other than the intended recipient — such as the chip owner, a regulator, or another secure processor chip in the system — from accessing or modifying the data in flight. This would be similar to existing confidential computing systems [4].

The tamper-proof enclosure would protect both the main chip and the secure processor from snooping or interference. If any attempt is made to tamper with the enclosure, a tamper-detection system would be triggered, activating mechanisms that will wipe any secret information on both chips and rendering the accelerator permanently inoperable by, e.g., blowing a large number of microscopic fuses on the accelerator chip.

The secure processor's firmware could be programmed to implement various rules and guarantees regarding the behavior of the accelerator. To enable multilateral governance, the secure processor would be configured to require each update of the firmware to be signed by a quorum (k of n) of authorized parties.

To ensure that all chips have the latest security updates and are subject to up-to-date rules, secure processors could be required to regularly — e.g., every three months — install a firmware update, or the secure processor will block the chip from operating. This update interval itself should be programmable so that it can be "tightened" or "loosened" depending on what a situation calls for. To prevent this from being used as a ratchet to implement arbitrarily intrusive rules, all chips could be allowed to revert to a permanently-approved firmware version implementing some baseline ruleset, such as a maximum cluster size rule as described in Appendix A.

The supply chain for the enclosures and secure processors would need to be secured and monitored to the satisfaction of all relevant parties to ensure that everyone can trust the integrity of the secure processors. While we will not discuss this in detail in this report, we believe that sufficient supply chain monitoring could be feasible through, e.g., random inspections and other measures.

Concrete Policy Options Enabled by FlexHEGs

This section will sketch how the flexHEG design could be used to implement and verify various policies. It is only a loose sketch, and is intended to be more illustrative than prescriptive or precise. Appendix A discusses potential mechanism implementations in more detail.

For any information going to and from the accelerator, the secure processor can:

- Verify the source and content of information coming in.
- Encrypt outgoing information to guarantee that only the intended recipient will be able to access it, and sign the information to allow the intended recipient to verify its source and integrity.
- Guarantee that no information leaves the system that is not supposed to.
- Monitor instructions going into the accelerator and track its behavior to ensure that any information it processes is processed as expected.
- Produce specific, verifiable claims about the state and behavior of the accelerator.

This combination of capabilities could allow the secure processor to verify the properties of complex workloads distributed across several accelerators by relying on encryption and cryptographic signatures to guarantee that information moving between the accelerators has not been spied on or modified in transit and by allowing the secure processors to make verifiable claims to each other about the behavior of their respective accelerators.

Verifying training run size: In order to determine the total amount of computing capacity that has gone toward training a particular model, each secure processor can keep track of how much computation has gone into producing various intermediate results and pass that information on to the next processor alongside the intermediate result itself. These "receipts" can then be traced back all the way to the randomly initialized starting point and summed to obtain the total number of floating-point operations that contributed to a final set of model weights. A very similar approach could be used to also account for the total amount of training data. [more]

Verifying other claims about training runs and other workloads: Along similar lines, the secure processors could also keep track of more sophisticated facts about what computations have and have not been performed during the training process, such as what kinds of tests were run on the weights during training or whether the training involved reinforcement learning or other types of feedback. This approach could also be generalized to other workloads performed on these accelerators, such as simulations and graphics processing. [more]

Privacy-preserving verification: These verified claims about workloads could then be reported to regulators, compute providers, or other stakeholders as needed. Secure processors could perform trustworthy "simplifications" on verified claims, e.g. by simplifying the claim "this model was trained in exactly this way" to "a model matching this hash was trained on a total of X FLOP", which could further be simplified to "a model matching this hash was trained with fewer than Y FLOP". This allows the reported verified claim to include the absolute minimum amount of information needed. [more]

Limiting training run size: An "interlock" that prevents the accelerator from executing instructions not marked as permitted by the secure processor would also allow proactive prevention of any violations of agreed-upon rules. For example, secure processors could refuse to permit an operation combining several intermediate results from other processors if doing so would result in the total number of floating-point operations spent on producing them to exceed some threshold. Certain devices owned by trusted entities could be provided with licenses that would allow them to go up to a higher limit than the default. [more]

Requiring evaluations: The rules could similarly stipulate more complex requirements, such as weights being subjected to certain types of evaluations or other checks at appropriate intervals during training. Such rules could only apply to models that are above a particular FLOP threshold. [more]

Controlling access to model weights: By controlling who can decrypt the resulting weights, the secure processor could also guarantee that, e.g., the model can only ever be decrypted and run on a specific set of flexHEG-capable inference chips, thus making the model radically more difficult to steal. The inference chips could also be required to implement certain rules in order to be allowed to decrypt and run the model. This would allow models to be widely distributed while still allowing that they are always used with certain safeguards in place. [more]

If rules are proactively implemented on-chip, this removes the need for any verification information to be shared with regulators or other parties.

Actually implementing these kinds of complex verification schemes in a robust way would be a significant software engineering challenge, but it should be an entirely solvable challenge, provided the secure processors offer a certain set of basic affordances combined with a flexible general-purpose processing capability.

If real-time, local verification and automated compliance guarantees as described above prove infeasible for particularly complex applications, verification could instead rely on after-the-fact analysis of verifiable "receipts" and hashes of intermediate results. Even this could likely be implemented in a way that keeps the details of the workload private. See Shavit [5] for further discussion of such methods.

Technical Objectives for FlexHEGs

In the list below, we propose a set of properties that a complete flexHEG design must have.[3] Next, we argue that a system with these properties would flexibly and securely enable all of the previously discussed guarantees and policy options. In the next section we describe a potential implementation of a system that has these properties and assess its technical feasibility.

Desired system properties:

1. **General-purpose logic** can be executed on-device.[4]
2. **Process to update** governance mechanisms if (and only if) the update has been approved by pre-specified (quorum of) stakeholders.
3. **System has access to accelerator data** needed for verification of compliance with a variety of rules.
4. **System can control accelerator** functioning for non-destructively enforcing regulations.
5. **Secure against non-invasive attacks** (e.g., the update process cannot be circumvented or the authorization keys cannot be stolen).
6. **Secure against physical tampering**. Ideally, circumvention by adversaries would not be technically feasible, even with physical access. A somewhat easier goal is to make circumvention so costly, per-device, that it is not worthwhile. Tamper-responsive designs can be complemented by randomized inspections for even greater assurance.
7. **Secure timekeeping** for implementing mechanisms that depend on the date or durations.
8. **Protection of confidential data** even if the device is compromised. For some threat models, a high per-device cost of extraction is a sufficient mitigation; for others, it would be beneficial to completely prevent data exfiltration even once.
9. **Authenticated and confidential inter-node communication** for governance coordination and protecting model weights (e.g., encrypting model weights or enforcing

[3] Although we are also interested in designs that compromise on some properties if that allows them to be deployed faster. For example, H100s could potentially be modified using post-processing to be tamper-evident, with firmware in existing processors updated to support various governance mechanisms (similar to [6]).

[4] For an ambitious version of automated governance, this general-purpose logic might be a large AI model itself.

a maximum number of accelerators that can simultaneously contribute to a single computation).
10. **Integrity of the system can be verified** to provide confidence to stakeholders.

Additional constraints:
11. **GPU cooling** is not significantly hampered.
12. **Manufacturability at scale in the near future** (2–4 years, or ideally sooner).
13. **Cost** of GPUs is not significantly increased.
14. **Efficiency** is not significantly decreased (performance, power consumption, and board space).
15. **Robustness against scalable attacks** against GPUs (e.g., tamper response cannot be maliciously triggered at scale or keys used to authorize updates cannot be deleted).
16. **Tolerable false-positive rate** (of tamper detection).
17. **Tolerable disruption of the maintenance** and replacement processes for GPUs.
18. **Acceptable integration with existing datacenter infrastructure** such as racks, cooling systems, and networking, i.e. the secure enclosure should not be too large to fit in a rack, or make it too difficult to connect the devices into a dense network fabric.

A system that meets constraints 11–17 could be integrated with accelerators with minimal disruption in the near future. If the system also has properties 1–10, then it would be able to measure accelerator usage, coordinate with other flexHEG-capable systems, perform checks on these data, and either credibly attest compliance or throttle usage if needed. These capabilities would be sufficient to implement all of the potential governance mechanisms discussed in the preceding section and Appendix A. It would also be able to support future governance mechanisms if they take the same form of observing the usage of one or multiple accelerators and then optionally performing rule checks. The system would also satisfy the strategic goals that it be able to update the governance logic, credibly demonstrate system integrity, and protect against a range of attacks.

Potential FlexHEG Design

A potential flexHEG mechanism could be built by modifying GPUs to include a secure processor for executing governance logic and a tamper-responsive enclosure to prevent physical tampering (as shown in Figure 1). Table 1 lists the sub-components of this potential system and why they are needed to achieve the design objectives. Next, we expand on their technical feasibility, design choices, and work that could advance each project.

Technical Component	**Motivation**
Tamper-responsive secure enclosure	Prevent and/or detect physical tampering.
Self-disablement mechanism	Disable the accelerator before governance mechanisms can be disabled if physical tampering is detected.
Secure processor	Securely execute logic for implementing guarantees.
Accelerator Interlock (input data and output controls)	Gather necessary information from GPU to verify compliance and be able to control the GPU for (non-destructive) enforcement.
Mechanism update process	Allow governance mechanisms to be updated if the changes are authorized by required parties.
Confidential, authenticated, and efficient communication between devices	Enable coordination between multiple flexHEG-capable devices (e.g., they can efficiently send secret weights or verify the authenticity of metadata).

Table 1: Motivation for technical components in a potential flexHEG design.

The flexHEG design discussed here is one of several options that could plausibly satisfy most of the technical objectives. Some alternative options for guaranteeable chips are:

- Modifying the firmware of existing accelerators to perform logic for implementing guarantees, and potentially also adding a tamper-evident or tamper-responsive enclosure with an additional assembly process.
- Integrating a secure processor directly into the accelerator chip either as a separate IP block or by obfuscating it within the rest of the accelerator circuits. This option could potentially be protected with a chip-scale secure enclosure instead of a larger PCB-scale enclosure.
- Building a bolt-on device that could be added to existing accelerators to monitor power usage and other high-level metrics.
- Integrating governance mechanisms on network switches or controller CPUs to govern clusters of GPUs in a server (with a server-sized secure enclosure).

We focus on this specific version because:

1. Including a flexHEG mechanism on every accelerator is the simplest way to ensure that each accelerator implements the guarantees (although there may be more efficient approaches).
2. A separate chip for the secure processor with unambiguous access to communication channels is likely easier to audit and trust than an integrated secure processor within the

closed-source accelerator chip (especially for actors with little visibility into the design process or supply chain). Using a separate chip is also likely less disruptive to the accelerator design process and enables reuse between different accelerators. However, a major potential downside of having the secure processor on a separate chip is that supply chain interception or tampering could potentially be used to obtain unrestricted accelerator chips.[5]

3. A tamper-responsive secure enclosure can be used in situations where tamper evidence alone would be insufficient (e.g., if conventional inspections or enforcement are not feasible).
4. Having access to extensive communications, memory, and meter data enables more sophisticated governance rules than are possible with high-level metrics like power usage.

However, the best type of design for implementing flexHEGs depends on the setting they are to be used in. For some settings, some of the more difficult technical requirements could be relaxed (especially if the flexHEG system has to be deployed before there is time to design and manufacture a new accelerator generation).

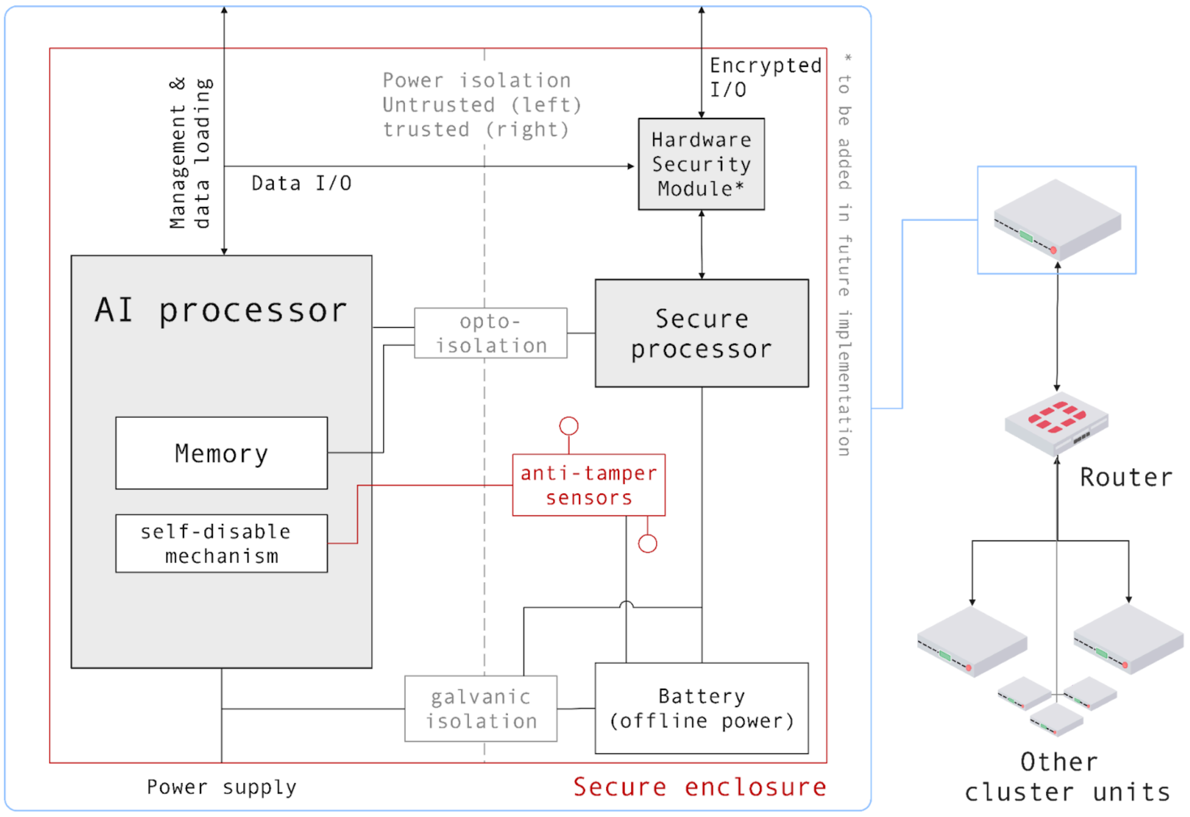

Figure 1: Potential system design with secure processor and tamper-responsive enclosure.

[5] Unless the accelerator chip is cryptographically paired so that it cannot operate without a particular secure processor.

Technical Components

This section focuses on the analysis of technical components needed for a potential flexHEG design: secure enclosure, self-destruct mechanism, secure processor, accelerator interlock, mechanism update process and interconnect encryption. In each subsection, we summarize the purpose of the component and explore promising ways to implement it.

Secure Enclosure

Summary: Secure enclosures have been used for over 20 years to defend cryptographic coprocessors and other chips from physical tampering [7], [8]. Secure enclosures could similarly protect GPUs if they can be modified to handle higher heat transfer, a larger temperature range, and GPU-specific input/output ports.

Design Choices

More advanced secure enclosures use active sensing to detect if the enclosure has been breached and then to trigger an appropriate response (in this case, some form of self-destruct). One approach for active sensing is to measure the capacitance of a serpentine-patterned conductor embedded around the enclosure [9]. Similarly, the resistance of a conductor mesh can also be measured [10]. If the mesh is disrupted, then the measurements will change, alerting the tamper-detection system. Measurements of the radio response function within an enclosure[6] could also be used [11] (although detection tests for the referenced prototype have so far only been done with metallic probes).

To ensure system integrity over the lifetime of the device, a battery inside or outside of the enclosure[7] can be used so that the sensors continue to operate even if external power is disconnected.[8]

Additional sensors can be used to prevent more sophisticated attacks by measuring for radiation, voltage glitching, lasers, temperature, and the rate of temperature change. To avoid having a single point of failure that could be targeted by an attacker, a distributed network of sensors could be used to independently trigger the tamper-response mechanism.

Larger enclosures that contain more GPUs are likely faster to design and deploy because they do not need to be as tightly integrated with the specific GPU.[9] However, enclosing multiple GPUs has a couple potential downsides: 1) GPUs frequently fail, and maintenance could be very

[6] This enclosure could potentially be made of metallized foam to enable airflow without providing a path for instruments to enter through (and the geometry of the metallized foam could potentially be sensed by the radio so that modifications could be detected).
[7] Having the battery outside the enclosure may be preferable if space is limited or if the battery has to be replaced.
[8] If power from the battery is also lost, then the tamper response is triggered.
[9] Data center security is an extreme extension of a large secure enclosure, which could be very important for some governance strategies (but is out of scope here because of our focus on technology R&D).

complicated while maintaining security; 2) the economics are less favorable because an attacker only needs to disable one enclosure to access multiple GPUs.

The FIPS-140-3 level 4 certification process [12] measures the physical defenses of tamper-responsive enclosures for cryptographic coprocessors. A modified version of the FIPS-140-3 certification process might be useful for verifying the security of completed flexHEG-capable systems.

At the time of this writing, devices that are certified at FIPS-140-3 level 4 [13] (the highest FIPS hardware-security rating) are sold by IBM [14] and by Private Machines [15]. One version of this project might be to modify existing secure enclosures to work with GPUs or to demonstrate why it would be feasible to modify these types of systems. This feasibility analysis could be performed by addressing how to adjust for the following differences between the use case with GPUs and cryptographic coprocessors:

1. GPUs have to dissipate much more heat;
2. GPU interconnect connections may be more extensive or delicate;
3. GPUs can reach high temperatures, which may change the tamper-sensor calibration.

GPUs are currently either air-cooled or liquid-cooled, and the trend is moving more towards liquid cooling (although it can be difficult to set up). Designs for secure enclosures will need to accommodate large amounts of heat transfer that may be necessary for future GPUs. This could potentially be done with a within-enclosure liquid cooling system and a liquid–liquid heat-exchanger at the enclosure boundary. Alternatively, the secure enclosure could be made of compressed metallized foam whose particular geometry is sensed by radio, and high pressure air flow could be used for cooling. As another option, the secure enclosure could potentially be made sufficiently thermally conductive and connected directly to the chip so that it can be cooled from outside the enclosure.[10]

Tamper-evident enclosures aim to solve a different version of this problem, where the goal is to be able to detect afterwards if an enclosure has been breached. Building tamper-evident enclosures could potentially be faster and simpler than tamper-responsive enclosures (they probably do not require a battery or self-destruct mechanism). However, tamper evidence is only useful in scenarios where GPUs can be inspected and where misuse does not need to be prevented immediately.

Technical Feasibility

It will likely be possible to modify secure enclosures to handle heat requirements, although thermal calculations for specific designs are needed. Based on the public literature, it is very difficult to circumvent the best secure enclosures. However, it is unclear whether nation-state

[10] Although, this would require the secure enclosure to be both quite thin and thermally conductive. E.g., even with a 2 cm thick copper enclosure, the temperature would reach 62° C to dissipate 700W from a die with an area of 820 mm^2. From https://en.wikipedia.org/wiki/Thermal_conductivity_and_resistivity: q = -k (T2 - T1)/L -> T1 - T2 = q*L/k = 700/(820*10^-6)*0.02/401 = 42° C temperature difference. So, if the outside temperature were 20° C, the die temperature would be 62° C.

attackers can compromise the best secure enclosures and, if they could, what the marginal per-device cost of circumvention would be.

Self-Disablement Mechanism

Summary: A self-disablement mechanism allows the system to become inoperable if an attempt to breach the secure enclosure is detected (and before the guarantee or defense logic can be disabled), thereby reliability preventing misuse of the accelerator.[11]

Design Choices

There are multiple approaches that could be used to disable a GPU. We favor methods that do not introduce significant novel safety hazards, are easy to manufacture, and are difficult for an adversary to disable.

One potential approach is to integrate thousands or millions of One-Time Programmable (OTP) memory bits throughout the GPU and modify individual GPU components (e.g., multiplication circuitry) to not function if the OTP memory has been set. If a tampering-detection signal is sent (or if there is a loss of power), then nearby capacitors could provide the energy to set the OTP bits. If we could be confident that there is no scalable way to reset these OTP bits, this would make for a reliable self-destruct mechanism.

Antifuses are an OTP component that are relatively simple to manufacture and cannot be reset by depositing electrons [16]. Placing them far from the surface of the chip would afford additional security. However, more thorough research would be valuable to compare potential OTP components and ensure that the one chosen is not susceptible to any scalable attacks (e.g., can antifuses be quickly repaired by an automated focused ion beam?).

Self-disablement mechanisms that do not require changing the accelerator chip design would be preferable because they could be added later in the design phase. One way to do this is by adding a thermal self-destruct mechanism using a material like nanothermite [17] in the packaging process. An additional benefit of thermal self-disablement is that the chip is unambiguously disabled, which also may make it more difficult for adversaries to reverse-engineer the design. However, using reactive materials like nanothermite may make manufacturing more complicated, and if improperly designed there may be safety concerns.

Technical Feasibility

A self-destruct mechanism based on OTP memory would likely be possible to manufacture, although additional work is needed to design the specific circuits. Other self-destruct mechanisms may also work, but challenges with manufacturing would need to be addressed.

[11] Although, some scenarios may not require self-destruct if it is sufficient to detect misuse via inspections for tamper-evidence without immediately preventing it.

Secure Processor

Summary: The purpose of the secure processor is to execute general-purpose logic for implementing guarantees.

Design Choices

The requirements for the secure processor are that it:

- Be secure
    - Without flaws or debugging logic that could be exploited;[12]
    - With firmware verification and rollback protection (See Mechanism Update Process).
- Have the appropriate inputs to monitor GPU usage (See Accelerator Interlock).
- Be performant enough to keep up with GPU usage.
- Have a real-time clock for time-dependent guarantees.
- Have access to a random number generator and side channel-resistant cryptography hardware (See Encrypted Interconnect).

There are many existing processor designs that could be used to satisfy at least a minimal version of these requirements, including a range of TPM cryptoprocessors [18]. As an example of a related (although closed-source and obfuscated) processor, the Intel Management Engine is included on many Intel CPUs for features like anti-theft prevention and capability licensing [19]. There are also open-source alternatives [20], [21], [22].

Additionally, we want various parties to trust that the logic is being executed as expected. To achieve this, we need to guarantee that no backdoors could have been added to the secure processor design or manufacturing process [23]. Some options that could reduce concerns about backdoors are:

- Open-sourcing the secure processor design and guarantee logic, which could then be analyzed and compared with physical scans of randomly selected chips.
- Executing some or all of the logic on FPGAs, which can have their configurations publicly audited, might be easier to check for physical inconsistencies because of their more uniform structure, and can use variants of configurations with equivalent logic to make many hardware trojans impractical [24].
- Sourcing multiple secure processors from different manufacturers[13] and running the same instructions on all of them before checking their outputs for consistency on-device (this approach is similar to the Lock-Step, which is used to provide fault tolerance in functional safety designs [25]). With three secure processors, they could detect disagreements and use ⅔ voting to choose which action to take.

[12] Formal verification may be useful here.

[13] Although this may increase the risk of a single compromised secure processor being used to sabotage chip usage.

There are also manufacturing techniques that would probably make tampering even harder (e.g., putting the secure processor near sensitive components like tamper sensors or interconnect channels).

As a faster but less secure option, a flexHEG mechanism could potentially be implemented on current GPUs by modifying the firmware of existing processors (similar to [6]).

Technical Feasibility

A secure processor could be built using an off-the-shelf design for a secure processor. Additional work may identify ways to make the processor more secure or trustworthy.

Accelerator Interlock

Summary: In order to verify adherence to arbitrary rules, the secure processor must be able to access all information relevant to making the compliance judgment. Similarly, the secure processor must be able to block GPU usage in the case that the compliance check was unsuccessful, thereby enabling non-destructive enforcement.

Design Choices

Ideally, the secure processor can access the following information:

- GPU instructions;
- Input/Output data;
- Memory snapshots;
- Usage metrics.

However, if the GPU IP is secret and/or too complicated to check over, we may not be able to fully trust the data it reports. At a minimum, information entering or leaving the chip could be routed through the secure processor, and clock cycles within the secure processor could be trusted.

It would also be useful to be able to write directly to memory, because some guarantees could rely on operations like randomizing the initial training weights.[14]

More ambitious guarantees might require a large AI model to be run locally to perform automated checks. This may require use of the AI accelerator to run the model, which could be difficult to do if the accelerator is not fully trusted. One approach to resolve this might be for the secure processor to randomly check some intermediate calculations performed by the accelerator.

[14] And it would be better to avoid trying to design checks to see if the data have been properly randomized.

If possible, it could be useful to redundantly measure important information and then perform consistency checks so that tampering with inputs is non-trivial. It may also be useful for security to obfuscate the information collection system.

Technical Feasibility

It would likely be possible to route inputs and outputs through buffers that the secure processor can access, although the cost of doing this could depend on GPU architecture details. Further work could identify other information that can also reliably be observed or controls that could be reliably applied.

Mechanism Update Process

Summary: For the guarantee logic to be flexible, it must be possible to update it on deployed devices. However, to prevent attackers from disabling the guarantees, the secure processor should only accept updates that were authorized by the appropriate parties.

Design Choices

This authorized firmware update process is typically implemented using public key cryptography, where the new logic must be cryptographically signed using the appropriate private key(s) for it to be accepted by the device.

The standard approach is to publish cryptographically signed updates and allow device operators to download and install them if/when they choose. A downside to this standard approach is that if the guarantees need to be modified to respond to new threats, device operators could simply not install the updated guarantees. If the flexHEG system is only being used to provide credible attestation about how the device is used, this would not be a problem (because the attestation would communicate which set of rules it is for). However, if the flexHEG system was needed to guarantee compliance with the updated rules, voluntary updates would be insufficient.

An alternative approach that could be used to enforce updates is to require that GPUs periodically receive a license specifying the minimum allowed version number. The secure processor could prevent usage of the GPU if it has operated for too long (as measured by a secure real-time clock or cycle counter[15]) without receiving a license, or if the minimum version number is greater than the current version number. The licenses and updates can be delivered by any channel, so no internet connection is required for this approach (which could be important for chips that are air-gapped for security).

The length of time that a license is valid affects the latency at which new updates must be loaded (e.g., if devices need a new license every three months, they could ignore an update for that much time). This pushes in the direction of making the license renewal time shorter so that

[15] If using a cycle counter, there is a risk that the operator secretly does not use the chip to save up usage time. This risk could potentially be mitigated by requiring usage reports.

guarantee changes are not overly delayed. However, with very frequent license renewals, the burden on operators becomes more significant. One approach that could be used to resolve this is to use data diodes [26], which could be used to transmit licenses without any risk of data leaving via the same path.

The update process could require that the update (or lack of update) be multilaterally approved by k of n parties, so that no single actor could unilaterally impose excessive restrictions. The list of which parties are allowed to authorize an update and how many votes are needed can also be modified in each preceding update.

Technical Feasibility

It would be possible to require that the secure processor regularly receives either approval or updated firmware signed by the correct parties in order to operate. Additional work could identify better alternatives or modifications that would allow the system to recover from a bad firmware update without compromising security.

Encrypted Interconnect

Summary:
Encrypted communication channels could allow secure processors to coordinate the implementation of guarantees across groups of GPUs operating together. Hardware cryptographic acceleration could enable all input and output traffic to be made confidential and authenticated.

Design Choices

Cryptography that provides confidentiality could allow the secure processor to choose specific GPUs that are allowed to read the outgoing data. Similarly, cryptography that verifies message authenticity could allow secure processors to trust that a particular message has come from another secure processor.

There are several potential uses of encrypted interconnect:

- Protect weights from being exfiltrated by an attacker snooping on interconnect;
- Constrain a training run to only use a certain, limited number of GPUs (similar to Fixed Set [3]) by only allowing specific GPUs to decrypt the information;
- Verify authenticity of metadata describing the compute graph, which could enable verification that training data were approved by a regulator or that model parameters have not exceeded an operation limit.

For low-throughput cryptography, a standard Hardware Security Module integrated with the secure processor would be sufficient. However, if the entire output stream through the interconnect needs to be encrypted (> 900GB/s [27]), substantially more cryptography hardware would be necessary.

NVIDIA's H100 already has dedicated AES-GCM hardware to encrypt PCIe traffic for single-GPU Confidential Computing [28]. For multi-GPU confidential computing, they will likely also need AES-GCM hardware for the NVLink ports to protect against an attacker snooping on the interconnect traffic.

Performing AES256 encryption without GCM authentication on all NVLink traffic would require around 3% of the total computing power of an H100.[16] If NVIDIA does not add the cryptography hardware to support confidential computing and the cost of adding it to the secure processor would be prohibitive, it may be possible to achieve some coordination between secure processors using occasional randomized authentication challenges.[17]

Technical Feasibility

NVIDIA's H100s already have AES-GCM acceleration hardware. The main open question is whether the efficiency/boardspace cost of encrypting all interconnect communications would be prohibitive. If it were prohibitive, there are more complicated/risky communication protocols based on randomized challenges that could be used for some of the use-cases.

Evaluation of Performance on Technical Objectives

Having outlined a potential flexHEG design, we will now evaluate how well it satisfies the technical objectives for flexHEGs. Table 2 contains a brief assessment of the potential design for each intended property or constraint.

Technical Objective	**Evaluation of Performance**
1. General-purpose logic for guarantees	Standard microprocessors or FPGAs could be used to perform general-purpose logic for guarantees.
2. Process to update guarantees	It is possible to require regular checks for updates and only allow updates that have cryptographically signed approval from k of n required parties. More detailed analysis would be useful to gain confidence about edge cases.

[16] Order-of-magnitude estimate for fraction of compute needed for AES-256 encryption of interconnect on an H100:
- H100 capable of 1.98 *10^15 int8 ops per second;
- Interconnect bandwidth of 900*10^9 bytes per second (source: https://en.wikipedia.org/wiki/Ampere_(microarchitecture));
- ~ 60 operations per byte for AES256 (source: Claude and wikipedia https://en.wikipedia.org/wiki/Advanced_Encryption_Standard);
- Total AES256 byte operations needed per second: 900*10^9 * 60 = 5.4*10^13;
- Fraction of total operations: 5.4*10^13 / (1.98 *10^15) = ~3% of GPU operations.

[17] However, a security model built around randomized challenges is likely more complex and susceptible to unexpected attacks.

3. System has access to GPU data	It is likely possible to access GPU instructions and communications without trusting the GPU. It is also likely possible to access usage metrics and memory, although how much these can be trusted is unclear.
4. System can control GPU	It is possible to enforce rules by preventing external communications or by throttling power access. Other methods could potentially be combined to improve robustness.
5. Secure against non-invasive attacks	Security against non-invasive attacks will depend on specific implementation details, although red-teaming, formal verification, and bug bounties can help identify flaws.
6. Secure from physical tampering	It will likely be possible to modify existing secure enclosures to work with GPUs. These enclosures are very complicated to circumvent, but the difficulty for nation-states is currently unclear.
7. Secure timekeeping	The secure processor will already be battery powered with sensors to detect side-channel attacks, which would also protect the clock.
8. Protect secret data	As with FIPS-140 level 4 cryptoprocessors, secret keys can be stored in protected registers with battery power to ensure persistence. If tampering is detected, these keys can be deleted.
9. Authenticated and confidential communication	Some accelerators already have AES-GCM hardware for encrypting PCIe traffic. It would likely be possible to add enough AES-GCM hardware to encrypt all traffic, and this may already be needed to enable multi-GPU confidential compute.
10. Integrity of system can be verified	This is potentially possible with FPGAs or open source secure processors designed by different actors.
11. GPU cooling	It is likely possible to modify secure enclosures to accommodate high heat transfer. More analysis is needed to choose a design with minimal security drawbacks.
12. Manufacturable at scale in near future	The secure processor and an OTP-based self-destruct mechanism can be manufactured with standard processes. Secure enclosures have been manufactured before, so it is likely possible to use similar manufacturing techniques.
13. Cost of GPUs is not significantly increased	The secure processor can likely be manufactured at low cost (relative to the accelerator). There are no major reasons to expect a high marginal cost for secure enclosures, but this will depend on

	the specific design.
14. Chip efficiency is not significantly decreased	The most significant potential computation is the encryption of interconnect traffic, which could use roughly 3% of total processing power.
15. Does not enable scalable attacks	Standard approaches for protecting private keys can be used to ensure licenses and updates can continue to be signed. Tamper sensors should be designed to be unsusceptible to scalable attacks.
16. Tolerable false-positive rate	This will depend on how the tamper sensors are calibrated. The false-positive rate should be thoroughly tested and prioritized as a design goal for the secure enclosure.
17. Tolerable disruption to maintenance	With one accelerator per enclosure, faulty accelerators can be swapped out in the same way they currently are.
18. Acceptable integration with infrastructure	If the secure enclosure geometry is designed to fit within existing servers then the impact on infrastructure will be minimal. More work on this is needed.

Table 2: Evaluation of the potential flexHEG design against the technical objectives for flexHEGs.

Future Work Towards FlexHEGs

In the previous sections, we proposed a set of criteria for a complete flexHEG system, along with a potential design for such a system. While our initial feasibility analysis found it likely that many of the objectives could be achieved, more work is needed to clarify specific design choices. The list below contains research questions that could make this flexHEG design more concrete and reduce technical risk.

i. What is a good physical design for a secure enclosure that allows for future heat transfer requirements and interconnect geometry?
ii. What type of OTP memory should be used, and what circuit design would be appropriate for distributed OTP self-disablement? (Or alternatively, is there a self-destruct mechanism that is superior to OTP disablement?)
iii. Which specific processor architecture(s) should be used for the secure processor? Are there good reasons not to use three redundant FPGAs, with a ⅔ voting scheme and error detection?
iv. Are there ways to legibly increase confidence that the flexHEG design does not allow for backdoors and that other parties' flexHEG devices are functioning as intended?
v. What are the potential failure modes of the proposed firmware update process and how can they be mitigated? (E.g., faulty update, private key stolen or deleted, etc.) Which cryptographic algorithms should be used? What are the implications of being able to dynamically change which parties can vote to authorize a firmware update?

vi. What hardware would be needed for accelerated interconnect encryption, and how much board space/power would it require? How much latency would this introduce? If this would be excessively costly and it is not already planned for confidential compute, what is the right design for a more lightweight protocol? (E.g., using randomized challenges.)

vii. How should the secure processor, crypto hardware, and accelerator interact? What are the pros and cons of integrating all into one chip vs. retaining them as separate components? How can one securely "pair" components if separate? How can one best share access to crypto hardware between confidential compute and flexHEG uses?

viii. How trusted should the accelerator be? If the accelerator is needed to run an AI model to check guarantees, is it necessary to randomly check the accuracy of some steps by recomputing them with the secure processor? How might this work?

ix. How best to track multi-GPU calculations using a compute graph? How to statically describe the compute graph to the secure processor (or alternatively, dynamically generate the compute graph)? How to efficiently use encryption to guarantee that between-accelerator communication is occurring as expected? How to take memory snapshots and compactly record the compute graph history? How to efficiently combine the compute graph history for snapshots without double-counting operations?

x. How secure are the best current secure enclosures (e.g., FIPS-140-3 level 4)? How do we legibly assess the security of future flexHEG designs once they have been built?

xi. Are there reduced versions of flexHEGs that would be faster or simpler to deploy?

Acknowledgements

We would like to thank Andrew Critch, Eddie Jean, Erich Grunewald, Evan Miyazono, Gabriel Kulp, Jacob Lagerros, Mehmet Sencan, Péter Drótos, and Will Hodgkins for useful discussions and comments.

Bibliography

[1] G. Sastry *et al.*, "Computing Power and the Governance of Artificial Intelligence," Feb. 13, 2024, *arXiv*: arXiv:2402.08797. Accessed: Aug. 23, 2024. [Online]. Available: http://arxiv.org/abs/2402.08797

[2] O. Aarne, T. Fist, and C. Withers, "Secure, Governable Chips. Using On-Chip Mechanisms to Manage National Security Risks from AI & Advanced Computing," Jan. 2024, [Online]. Available: https://www.cnas.org/publications/reports/secure-governable-chips

[3] G. Kulp *et al.*, "Hardware-Enabled Governance Mechanisms: Developing Technical Solutions to Exempt Items Otherwise Classified Under Export Control Classification Numbers 3A090 and 4A090," RAND Corporation, Jan. 2024. Accessed: Apr. 09, 2024. [Online]. Available: https://www.rand.org/pubs/working_papers/WRA3056-1.html

[4] Confidential Computing Consortium, "A Technical Analysis of Confidential Computing," Nov. 2022, [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3_unlocked.pdf

[5] Y. Shavit, "What does it take to catch a Chinchilla? Verifying Rules on Large-Scale Neural Network Training via Compute Monitoring," May 30, 2023, *arXiv*: arXiv:2303.11341. Accessed: Aug. 23, 2024. [Online]. Available: http://arxiv.org/abs/2303.11341

[6] J. Petrie, "Near-Term Enforcement of AI Chip Export Controls Using A Firmware-Based Design for Offline Licensing," May 28, 2024, *arXiv*: arXiv:2404.18308. doi: 10.48550/arXiv.2404.18308. Available: https://arxiv.org/abs/2404.18308

[7] R. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems, 3rd Edition." Accessed: Aug. 23, 2024. [Online]. Available: https://www.wiley.com/en-us/Security+Engineering%3A+A+Guide+to+Building+Dependable+Distributed+Systems%2C+3rd+Edition-p-9781119642817

[8] J. Obermaier and V. Immler, "The Past, Present, and Future of Physical Security Enclosures: From Battery-Backed Monitoring to PUF-Based Inherent Security and Beyond," *J. Hardw. Syst. Secur.*, vol. 2, no. 4, pp. 289–296, Dec. 2018, doi: 10.1007/s41635-018-0045-2.

[9] H. Eren and L. D. Sandor, "Fringe-Effect Capacitive Proximity Sensors for Tamper Proof Enclosures," in *2005 Sensors for Industry Conference*, Feb. 2005, pp. 22–26. doi: 10.1109/SICON.2005.257863.

[10] P. Isaacs, T. M. Jr, M. J. Fisher, K. Cuthbert, and W. L. Gore, "TAMPER PROOF, TAMPER EVIDENT ENCRYPTION TECHNOLOGY," *Pan Pac. Symp.*, 2013, [Online]. Available: https://www.circuitinsight.com/pdf/tamper_proof_encryption_smta.pdf

[11] P. Staat, J. Tobisch, C. Zenger, and C. Paar, "Anti-Tamper Radio: System-Level Tamper Detection for Computing Systems," Dec. 16, 2021, *arXiv*: arXiv:2112.09014. Accessed: Aug. 23, 2024. [Online]. Available: http://arxiv.org/abs/2112.09014

[12] I. T. L. Computer Security Division, "FIPS-140-3: Cryptographic Module Validation Program | CSRC," CSRC | NIST. Accessed: Aug. 23, 2024. [Online]. Available: https://csrc.nist.gov/projects/cryptographic-module-validation-program

[13] I. T. L. Computer Security Division, "Search - Cryptographic Module Validation Program | CSRC | CSRC," CSRC | NIST. Accessed: Aug. 23, 2024. [Online]. Available: https://content.csrc.e1a.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search?SearchMode=Advanced&CertificateStatus=Active&ValidationYear=0&OverallLevel=4

[14] "IBM: 4769-001 Cryptographic Coprocessor." Accessed: Aug. 23, 2024. [Online]. Available: https://www.ibm.com/docs/en/power9?topic=ad-4769-001-cryptographic-coprocessor-fc-ej35-ej37-bsc-ccin-c0af

[15] "ENFORCER™ Details | Private Machines Inc." Accessed: Aug. 23, 2024. [Online]. Available: https://privatemachines.com/enforcer-details/

[16] S.-Y. Chou, Y.-S. Chen, J.-H. Chang, Y.-D. Chih, and T.-Y. J. Chang, "11.3 A 10nm 32Kb low-voltage logic-compatible anti-fuse one-time-programmable memory with anti-tampering sensing scheme," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2017, pp. 200–201. doi: 10.1109/ISSCC.2017.7870330.

[17] G. Jian, S. Chowdhury, K. Sullivan, and M. R. Zachariah, "Nanothermite reactions: Is gas phase oxygen generation from the oxygen carrier an essential prerequisite to ignition?," *Combust. Flame*, vol. 160, no. 2, pp. 432–437, Feb. 2013, doi: 10.1016/j.combustflame.2012.09.009.

[18] "Secure Enclave," Apple Support. Accessed: Aug. 23, 2024. [Online]. Available: https://support.apple.com/en-gb/guide/security/sec59b0b31ff/web

[19] "What is Intel Management Engine?," Intel. Accessed: Aug. 23, 2024. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000008927/software/chipset-software.html

[20] "Open source silicon root of trust (RoT) | OpenTitan." Accessed: Aug. 23, 2024. [Online]. Available: https://opentitan.org/

[21] "Tropic Square | TROPIC01." Accessed: Aug. 23, 2024. [Online]. Available: https://tropicsquare.com/product

[22] B. Kelly *et al.*, "Caliptra: A Datacenter System on a Chip (SOC) Root of Trust (RoT)," Jul. 2022, Accessed: Aug. 23, 2024. [Online]. Available: https://www.opencompute.org/documents/caliptra-silicon-rot-services-09012022-pdf

[23] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An Overview of Hardware Security and Trust: Threats, Countermeasures, and Design Tools," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1010–1038, Jun. 2021, doi: 10.1109/TCAD.2020.3047976.

[24] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in FPGA devices: threat analysis and effective counter measures," in *Proceedings of the 24th edition of the great lakes symposium on VLSI*, in GLSVLSI '14. New York, NY, USA: Association for Computing Machinery, May 2014, pp. 287–292. doi: 10.1145/2591513.2591520.

[25] X. Iturbe, B. Venu, E. Ozer, and S. Das, "A Triple Core Lock-Step (TCLS) ARM® Cortex®-R5 Processor for Safety-Critical and Ultra-Reliable Applications," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, Jun. 2016, pp. 246–249. doi: 10.1109/DSN-W.2016.57.

[26] H. Okhravi and F. T. Sheldon, "Data diodes in support of trustworthy cyber infrastructure," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, in CSIIRW '10. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 1–4. doi: 10.1145/1852666.1852692.

[27] R. Merritt, "What Is NVLink?," NVIDIA Blog. Accessed: Aug. 23, 2024. [Online]. Available: https://blogs.nvidia.com/blog/what-is-nvidia-nvlink/

[28] G. Dhanuskodi *et al.*, "Creating the First Confidential GPUs: The team at NVIDIA brings confidentiality and integrity to user code and data for accelerated computing.," *Queue*, vol. 21, no. 4, pp. 68–93, Aug. 2023, doi: 10.1145/3623393.3623391.

[29] A. Ho, "Algorithmic Progress in Language Models," Epoch AI. Accessed: Aug. 23, 2024. [Online]. Available: https://epochai.org/blog/algorithmic-progress-in-language-models

Appendix A: A Deeper Dive into Possible Guarantees

This section will provide more detailed sketches of how flexHEGs could be used to implement a range of verification and automated compliance mechanisms. These are not intended to be a complete description, nor are they a strong recommendation of this exact approach. Rather, the goal is simply to illustrate what kinds of guarantees flexHEG mechanisms could enable.

We will begin by a simple maximum cluster size mechanism that could provide a simple baseline, allowing small clusters to be exempt from other types of rules. We will then sketch a basic approach to verifying claims about workloads before moving on to describing how workload verification could be extended to automated compliance. We will also describe how flexHEG would allow controls on how results, such as trained models, can be shared, and how this would improve security. We will then discuss some more complex schemes that would combine these capabilities to solve AI governance challenges.

Maximum Cluster Size as a Minimal Rule for Small-Scale Use Cases

Most of the verification and automated compliance mechanisms discussed in this report essentially require all workloads to be articulated in a format that the secure processor can analyze and verify claims about. This may be difficult to implement and unnecessarily restrictive in many cases. As the primary concerns addressable by guaranteeable chips relate to large-scale AI applications, chips that are limited to sufficiently small clusters could be left exempt from any rules and requirements. This could also create a desirable safeguard against overreach. We sketch a mechanism for implementing this, based on the "Fixed Set" mechanism described by Kulp et al. [3].

To implement an effective limit on maximum cluster size, the guaranteeable chips would, by default, only communicate with other chips at a very low bandwidth, making them difficult to use for workloads that do not fit on a single accelerator. If a user did want to use the chip as part of a small cluster, they would ask a set of guaranteeable chips to form a cluster with a set of other guaranteeable chips. The guaranteeable chips would all verify with each other that they are in agreement regarding which chips are part of their network and ensure that the network's total FLOP/s capacity falls under some particular threshold. Any traffic outside the network would only be allowed to occur at very low bandwidth[18] in order to guarantee that this cluster cannot be usefully combined with other clusters to run larger workloads. Exceptions could be made for data that are encrypted in such a way that they can only be decrypted on the same cluster. This would allow, e.g., fast off-cluster backups without concerns that the backups would actually be fed to a different cluster. The secure processor might also be configured to lift these limits if the chip is running a trusted workload that has been signed by a regulator, such as inference on a known-to-be-safe model.

The maximum allowable size for these rule-free clusters should likely be determined relative to a particular threshold at which more demanding guarantees would be required. For example, if models trained above 10^k FLOP would be considered potentially dangerous and required to be trained in particular ways, the maximum allowable size for rule-free clusters could be set at some level such that it would be infeasible to attempt a 10^k FLOP training run using a large number of these rule-free clusters. This would depend on making some modeling assumptions regarding the penalty that would be imposed by the limited communication bandwidth

[18] This bandwidth limit might be articulated in terms of an average over a period of several days, allowing data to be moved onto and off the cluster relatively quickly at the beginning and end of a workload, while preventing workloads that would require sustained high-bandwidth communication.

between the rule-free clusters. However, this threshold should be set conservatively to account for the fact that algorithmic innovations would likely reduce the amount of compute required to attain some particular, potentially dangerous level of capabilities [29].

Alternatively, the maximum allowable size for rule-free clusters could be set relative to the capabilities of non-guaranteeable compute available at the time. For example, if there are already large numbers of non-guaranteeable chips in the world such that it would be infeasible to prevent anyone from building a 10^x FLOP/s cluster using them, there would be relatively little purpose in setting the maximum allowable size for rule-free clusters at any level below 10^x FLOP/s.

Verifying Claims about Workloads

Put in general terms, various computing workloads performed on AI accelerators can be thought of as taking some data and processing them in some way. Generic data analysis typically involves taking some data, filtering and transforming them in some way, and calculating statistics over the resulting "cleaned" data. Training a machine learning model typically looks like taking randomly initialized weights and a dataset and processing them together to modify the weights to perform better on some task. Running a simulation will involve taking some starting state and processing it according to some set of rules to obtain the next state of the simulation, and so on.

Secure processors could be used to verify various types of workloads by producing verifiable "receipts" of what data were processed and of how, resulting in a description of how some final result (such as a trained model) was produced. Secure processors could aggregate and analyze these receipts and produce higher-level receipts that make more abstraction claims about the workload. These higher-level receipts could include only the minimal amount of information that needs to be reported, e.g., that a particular model was trained with a total amount of FLOP that does or does not exceed some threshold.

For example, in order to determine the total amount of computing capacity that has gone toward training a particular model, each secure processor can keep track of how much computation has gone into producing various intermediate results, such as activations, gradients, and shards of weights, and pass that information to the next processor in a "receipt" accompanying the intermediate result itself. These receipts can then be traced back all the way to the randomly initialized starting point and summed to obtain the total number of floating-point operations that contributed to a final set of model weights.

Receipts could also be combined to make claims about what has *not* been done by combining receipts describing what has been done. For example, someone who operates a large number of AI accelerators for the purposes of running simulations unrelated to AI could aggregate receipts describing all of their simulation workloads from a particular month to produce a receipt stating "this set of chips was not used for AI workloads during this particular month." If some accelerators were simply idle or offline, the secure processor would be able to verify this as well.

This description is simplified and overlooks some open problems, such as how to ensure that additional compute is not implicitly smuggled in, e.g., through inputs that are supposedly training data. However, these appear to be likely solvable, e.g., by requiring developers to keep verifiable records of which data they trained on so that the data can be inspected (again, ideally in a privacy-preserving manner) to check for foul play.

Automated Compliance with Rules

If the secure processors are continuously tracking and generating "receipts" throughout the workload, this could also be used to automatically guarantee compliance with constraints on workloads.

For example, a rule limiting AI training workloads to a maximum compute threshold could be enforced by having the secure processor block any operation that would result in a training run exceeding this threshold, e.g. as a result of the operation merging or further processing previous intermediate results.

Alternatively (or complementarily), the secure processors could perform static checks on a description of the workload before it begins to make sure it complies with all rules, and then monitor the behavior of the involved accelerators to check that the actual computation matches the plan.

This kind of automated compliance could be preferable to verification because it would remove the need for the chip owner to communicate anything at all to the regulator and would prevent anyone from breaking rules even if they are indifferent to later punishment. Making fully automated compliance reliable would likely be a greater software engineering challenge. One advantage of a verification-focused approach is that the chip operator can play a part in figuring out how they can structure their workload such that they can produce the required verification of compliance, whereas a ruleset-based approach would require a regulator to write a fully general compliance system that could analyze any future workload and determine whether it is in violation.

Controlling the Sharing and Deployment of Results

Restrictions on how the results of some computations can be shared could also be implemented using flexHEG mechanisms. For example, if an AI model has been trained with an amount of compute exceeding some threshold, the secure processor could refuse to allow the weights to be moved off the chip unless they are encrypted such that they can only be decrypted by other guaranteeable chips.

In some cases, the weights might be decryptable by any other flexHEG-equipped chip. In this case, the purpose of this restriction would be to guarantee that the weights cannot be further modified without approval and that the deployment of the model is compliant with any deployment-related rules that all flexHEG chips implement. This would allow models to be released to be used freely by anyone, while still guaranteeing that any safeguards integrated into the model remain in place.

More commonly, a company might encrypt a model such that only specific chips owned by their partners or customers can run the model. This would allow a model to be widely deployed, while making it much more difficult for anyone to steal the weights.

More Complex Rules and Verified Claims

Required Evaluations

This section has so far focused primarily on claims about the quantity of compute used, but this is of course not the only model property of interest for governance. Training compute is merely a proxy for the overall capabilities of the model.

Potentially, a flexHEG could be used to require that, when a given compute threshold is reached, the model must be subject to a particular automated capabilities evaluation before training can continue or before the model can be deployed. An adaptive set of follow-up actions could be required depending on the result of the evaluation. For example, if the model does not appear to have any dangerous capabilities, further training or deployment could commence as normal. If the evaluation suggests that the model *does* have dangerous capabilities, it could be required to be subjected to further evaluations or safety and security measures before further training or deployment.

To govern deployment, flexHEG-capable inference chips would require a receipt confirming that any deployed models above some size or compute threshold have passed certain evaluations, or the chip will refuse to run the model.

Mutually Private Evaluations

Following the open source approach of the flexHEG design, the automated evaluations discussed in the preceding section would of course be open source and inspectable by the developer. However, in some cases it may be desirable to keep an evaluation suite hidden from the AI developer, e.g. to prevent "teaching to the test", or to protect proprietary IP of the evaluation provider. FlexHEG mechanisms could allow evaluations to be run while keeping the evaluation suite secret from the model developer, and keeping the model weights secret from the evaluation developer. A secure processor could be set up to run a program that applies some evaluation to a model by running some function that takes the model as input, and provides e.g. a binary pass/fail result. The evaluation developer would provide the function, encrypted such that only the secure processor can decrypt it. The model developer would provide the model weights and architecture, similarly encrypted. Both sides could verify in advance that the secure processor's program will only run the evaluation function, and will only release the result of the evaluation, and will never release either the model weights or the evaluation function outside the device.

Required Sharing of Claims or Results

Controls on the sharing and deployment of models, as described above, could also be used to guarantee that developers will share certain information with other parties. The secure processors could prevent a set of weights from being moved off a cluster, deployed, or (optionally) trained further until they have received cryptographically signed confirmation from some other party that certain information has been received. For example, developers could be required to share certain evaluation results with regulators before being able to further train or deploy a model. This kind of mechanism could potentially also be used to guarantee that after a model has finished training, the weights will be shared with other stakeholders, such as other state parties to some treaty, or simply investors.

Limitations of These Mechanisms

This approach relies on certain assumptions about AI systems, such as that the most powerful and governance-worthy systems are most compute intensive. These mechanisms also rely on even more specific assumptions, such that any single AI system will depend on high-bandwidth communication between its components. There will likely be some kinds of AI systems that violate these assumptions, and more will be discovered as various actors attempt to game the mechanisms.

Mixture of Experts (MoE) systems are one example of existing systems that violate some of our assumptions and could be used to partly circumvent these mechanisms. An MoE system essentially consists of several different neural networks that are each trained to handle a particular subset of possible inputs well. Any input that the system receives is then “routed” to one or more of these experts. Because there is no significant amount of communication between the experts, guaranteeable chips would have no straightforward way of knowing that a neural network they are running or training is not in fact the entire system, but merely one of the experts in a larger system. This would allow limits on system size to be partly circumvented. However, making the individual experts smaller will harm the performance of an MoE system and, intuitively, should limit the maximum level of intelligence the system can exhibit. This means that system size limits could still be effective if they are adjusted to appropriately account for circumvention strategies such as Mixtures of Experts.

Further research is needed to identify and assess different circumvention strategies like this. The updateable nature of the flexHEG design fortunately means that rules can be updated to account for newly discovered circumvention strategies.

This limitation of the flexHEG design may generally push AI developers toward more modular systems, consisting of several networks with more limited communication between them. This may actually prove to be desirable for reducing risks from AI: Modular models would likely be easier for humans to interpret, reason about, and control, and could make it easier to ensure that the models have desirable safety characteristics.

If necessary, FlexHEGs could also be combined with other forms of oversight to potentially address some issues like this. For example, to address worries about MoE systems, developers could be required to report basic metadata about their training runs to regulators. The regulator could then review these to look for signs of efforts to circumvent rules, such as large numbers of similar training runs being run at the same time. Requiring developers to report what each model is intended to be used for would also make it more difficult to hide these systems.

Appendix B: Concerns about Privacy, Misuse, and Security

To enable international coordination based on strong mutual assurances, it might be desirable that, eventually, all powerful AI accelerators will be equipped with flexHEG mechanisms. This prospect will, very understandably, raise concerns in the minds of many readers. In this section, we hope to discuss how our proposal avoids many concerns and how some remaining concerns might be mitigated. We first discuss concerns about the use of flexHEG for government surveillance and control before moving on to concerns about the secure processor creating vulnerabilities for attackers to exploit.

Government Surveillance and Control

First, we want to emphasize that, in order to achieve the described benefits for international coordination and assurance, these mechanisms would need to be added to powerful AI accelerators used in datacenters, not on anyone's personal devices. Therefore, they would have limited usefulness for control of, e.g., ordinary political activities.

Additionally, these mechanisms would not allow governments to unilaterally spy on anyone's chips or to intervene in their operation on an arbitrary, ad hoc basis. Nonetheless, verification and automated compliance capabilities could still, in principle, be used to implement more efficient and effective mass surveillance and control. Note, however, that this would not be qualitatively different from what a motivated government could achieve with less technically sophisticated means. For example, governments could relatively easily adopt a policy of only allowing powerful AI chips to be purchased by licensed cloud providers and requiring these cloud providers to surveil and control what their customers do with the chips.

Several measures could be taken to address concerns about overreach.

- Governments could ideally clear legal, and perhaps even constitutional, limits on the nature of the rules implemented through flexHEGs to ensure they do not encroach on civil liberties or otherwise exert excessive control.
- The mechanisms could be governed by an international process that is designed to ensure that these mechanisms are only used to implement rules that are necessary for

international security. The devices would refuse updates that have not been signed by a quorum of parties to this process.

- Instead of requiring all devices to be up to date, moving to new rule sets could perhaps be incentivized by having other flexHEG devices refuse to work with devices that are out-of-date relative to themselves, including e.g. refusing to send some frontier models to such chips for inference. However, it is unclear whether this would create sufficient incentives to keep the system effective.
- The mechanisms could be configured to irrevocably accept a roll-back to some minimal baseline rule set such as one that would restrict the total size of clusters, and nothing else. We discuss how a maximum cluster size rule could be implemented in Appendix A.

Indeed, if AI development comes to be seen as a matter with significant implications for international security, comparable to e.g. nuclear technology, various countries would likely attempt to monitor and control it in various ways. By allowing rules to be implemented locally and automatically, flexHEGs would likely represent an option that would preserve privacy, freedom, and the rule of law more effectively than more ad hoc, lower tech alternatives, such as monitoring, extensive reporting requirements, physical inspections, and tight restrictions on who is allowed to buy or access powerful chips.

Third Party Abuse

Adding an additional processor to a system will inevitably add complexity and thus create some additional attack surface. The secure processor may appear particularly concerning from this perspective because it has, by design, extensive access to the accelerator.

To address this, the secure processor should be designed extremely carefully, and the firmware loaded to it would need to be extremely thoroughly tested and verified. It should likely be based on a maximally simple, well-vetted processor design. The standardized, open-source nature of the system would also allow anyone to study the system to help find possible vulnerabilities, improving security. The required firmware update mechanism would enable issues to be addressed if any vulnerabilities were later found.

Importantly, the secure processor would not rely on "phoning home" to anyone and would only need to receive and send very specific information outside the system. This means that guaranteeable chips could realistically be kept in an air-gapped environment, with only occasional firmware updates and possible license keys needing to be brought in to the outside, and some signed claims being moved out of the system, if needed.

The addition of the secure processor also has several benefits for security:

- The enclosure would protect against many side-channel and fault-injection attacks that AI accelerator chips are currently vulnerable to.
- The encryption capabilities of the secure processor would allow any significant traffic between devices to be encrypted in flight. For example, the secure processor could be configured to never release key intellectual property, such as frontier model weights,

outside the system unless encrypted in such a form that it can only be decrypted by other secure processors that will implement the same policy. This could significantly mitigate, e.g., insider threats.

- Because it sits directly between the accelerator and the outside world, the secure processor could also act as a barrier and could be used to implement safeguards such as the automatic detection of suspicious instructions that may be attempting to attack the accelerator.