BlindAI DRM with FLI: Secure monitoring of AI models deployed on-premise

Mithril Security

03.10.2023

Table of contents

Introduction

Problem statement

LLMs today show human-like capabilities in various tasks and seem to have the potential to achieve superhuman levels of intelligence.

While being extremely promising to help advance society, their capabilities and misuse could have dire consequences on society, from cyber threats to massive disinformation campaigns.

Monitoring and having an off button to disengage such models are potential ways to make sure to regulate the use of these models by potentially untrusted actors.

However, having technical proof that such mechanisms are enforced can be complex.

Fortunately, at Mithril Security, we have developed the technical expertise to leverage secure enclaves, a recent hardware-based technology, to provide cryptographic guarantees that models shared with untrusted parties can still be controlled remotely.

We will present in this document a Proof of Concept of BlindAI DRM, a system to provide remotely monitorable AI, where consumption of the AI model can be controlled and tracked by a third party with cryptographic guarantees.

Parties definition

In order to understand BlindAI DRM better, let's define the parties considered:

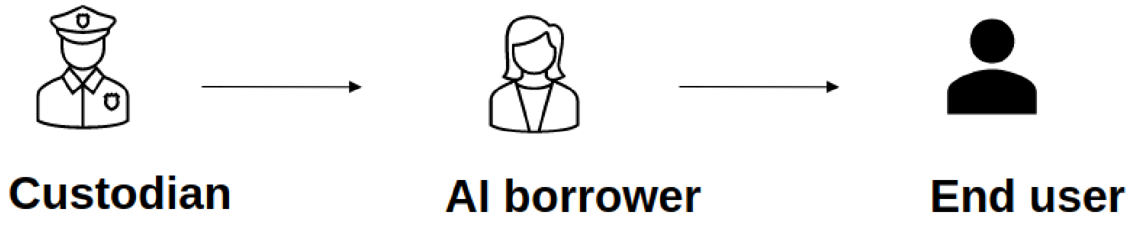

Figure 1: BlindAI DRM key actors

- **Custodian:** They provide the AI to the AI borrower and track and potentially block AI consumption
- **AI model borrower:** They borrow the custodian's AI, hosting it on their infra. They will benefit from the AI, but the AI provider doesn't want to share the weights with them directly, as they want to ensure they can only consume the model, and not reverse engineer it.
- **AI consumer:** The end user wishing to query the AI model. This could be the AI borrower, or it may be that the AI borrower resells the model to a separate end user.

Example: controlled AI consumption

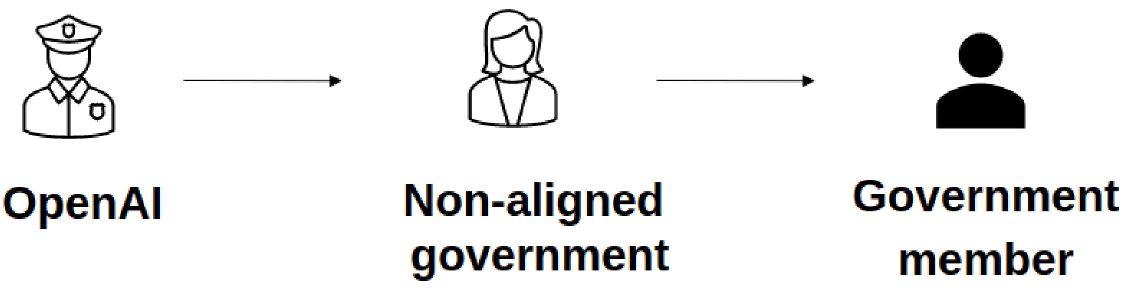

Figure 2: BlindAI DRM example actors

We could imagine a use case where we have:

- **OpenAI** with **GPT4** as the **AI custodian**
- **A country that is not aligned with the US as the AI borrower.** It would benefit from GPT4 for internal usage but should not be trusted with the IP of the weights directly.
- **A specific department or member of the government** as the **end user.**

The custodian wants to lend their model to the borrower who would consume it through a local deployment, without exposing the model weights to avoid uncontrolled usage and IP theft.

Objectives

The desired properties for the custodian in this scenario are:

- The weights of the model are never directly accessible to the AI consumer or AI model borrower, so as to avoid uncontrolled use of the model
- The custodian defines a computational budget, which is the number of queries the AI consumer can make to the borrowed AI model, before having to explicitly ask the custodian to extend their budget
- Without explicit approval of the custodian, the model is unusable. At any time, the custodian can hit a stop button to make the deployed model unusable.

Solution

We propose a solution called BlindAI DRM, which uses secure enclaves, which are hardware-based confidential and verifiable environments, to allow the custodian to share their model while ensuring all the security properties we mentioned regarding model weights protection.

More details about secure enclaves technical details, such as hardware requirements, confidentiality and integrity properties, are provided in the technical deep dive.

Main ideas

To answer the needs of this controlled AI usage, the custodian will ship a hardware enclave (in our case we use an Intel Xeon 3rd Gen processor), which contains a code with all the right restrictions, and whose properties will be enforced by the secure hardware.

The main job of the secure enclave is to:

- Ping the custodian infrastructure, and prove its identity and security properties, especially prove that the secure enclave has not been tampered with.
- Receive the model, sent by the custodian after cryptographic verification of the enclave, and then serve it.
- Ensure the custodian is always in control. The enclave acts as an AI DRM for the custodian.

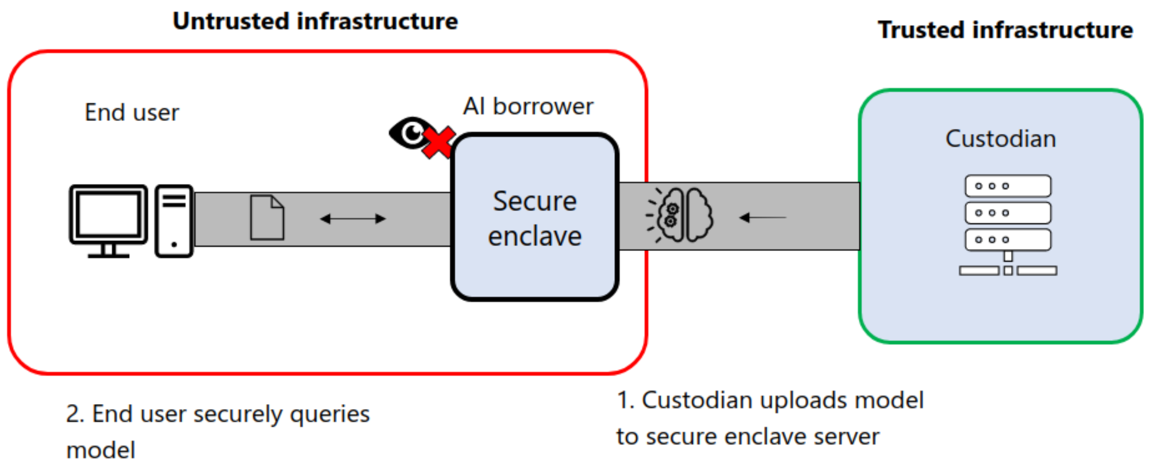

Figure 3: BlindAI DRM infrastructure

Components

BlindAI DRM is made up of three main components:

End user client:
The end user client server is used by AI consumers to query the model inside the enclave server.

Secure enclave server:
This server is deployed by AI borrowers and enables them to host the custodian's AI model without having access to the model weights, thanks to the use of secure enclaves.

Custodian server:
This is the server used by the custodian to:

- Securely upload the model to the enclave hosted by the AI borrower

- Follow the consumption of AI models
- Block all future consumption of AI with immediate effect by using a “stop switch”

Demo

A demo can be found here to see our project work in practice.

Workflow

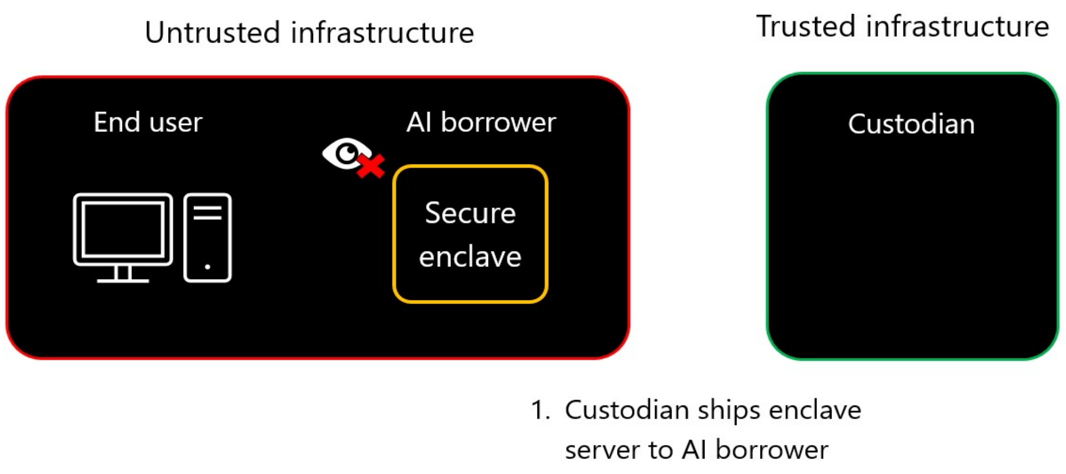

Figure 4: 9-step BlindAI DRM workflow animation

Let’s now walk through the controlled AI consumption workflow cycle from start to finish:

1. **Custodian ships enclave server:** The custodian will ship a server leveraging secure enclaves to the AI borrower. A secure enclave is a hardware-based highly confidential environment that will ensure the confidentiality of their weights.

2. **Custodian launches server:** The custodian then launches their server, which will wait for a connection attempt from the enclave server. They specify an AI computing budget for the end user with the number of requests the end user will initially be able to make.

3. **Enclave server sends attestation report to custodian server:** The enclave server is launched. It has no model preloaded, pings the custodian server and shares cryptographic attestation assets to prove its authenticity. At the end of attested TLS, which is detailed below, the custodian will know:
    a. The enclave has been loaded with the right code, and that the enclave will follow the DRM properties that the custodian expects.
    b. The enclave has the right security properties, especially hardware-based memory isolation to prevent reading memory to extract the model.

    c. A symmetric key has been negotiated and the custodian can send the model without anyone else knowing what was the data, as only the enclave has the decryption key, and no one is able to extract it from the enclave.

4. **Custodian server sends model:** Once the custodian has performed attested TLS, the model is securely sent to the enclave.

5. **End user query:** The end user can then request the enclave to consume the AI.

6. **Inference:** The enclave verifies if the end user is within their computing budget, and if so it performs inference and sends results back to the end user server.

7. **(Where applicable) Computing budget increase request:** If the end user has reached their computing budget threshold and wishes to make a new request, the enclave will send a request to the custodian server to extend the end user's AI computing budget.

8. **(Where applicable) Custodian response:** The custodian will approve or deny the request to unlock additional requests for the end user.

Manual "stop" switch

The custodian can choose to manually shut down the custodian server at any time, making the enclave server obsolete. Any further queries to the enclave will fail.

How to implement the BlindAI DRM MVP

In order to test our BlindAI DRM MVP, please refer to our GitHub repo which gives you full access to the code and step-by-step instructions for deploying the solution.

Security deep dive

To understand BlindAI DRM better, we will introduce you to the key underlying technologies behind the solution.

All our solutions are based on Confidential Computing, the field of using Trusted Execution Environments, also known as **secure enclaves**, to process sensitive data with transparency, even in memory.

All major hardware providers, Intel, AMD and Nvidia, propose Confidential Computing capabilities on their data center-grade processors with for instance Intel SGX, AMD SEV or Nvidia Confidential GPU.

Secure enclaves

What are secure enclaves?

Secure enclaves are **highly isolated compute environments** where data and applications can reside and run. Sensitive data sent remains encrypted at most times, and is only decrypted inside the enclave. Even actors with access to the host machine running the enclave will not be able to access the content of the enclave.

Secure enclaves offer:

- **Data confidentiality** from data at rest, transit and even in memory as read or write memory attempts will fail thanks to hardware isolation mechanisms.
- **Transparency** thanks to remote attestation, i.e. proof we are talking to a secure enclave with the expected code and security properties

How do they work in BlindAI?

With BlindAI DRM, AI is deployed in an **Intel SGX secure enclave**. SGX enclaves were the first ones to be available on the market. They are created within a **private region of memory** that is isolated from other processes running, even those with higher privilege levels.

External access is always denied. This means that even someone with access to the host infrastructure cannot access the model weights being served within the enclave. Thus, AI shipped in an enclave is protected from model theft.

Attestation

What is attestation?

Attestation is a process which allows us to have hardware-derived proof that we are communicating with the expected application and that it is running inside a secure enclave. It can be viewed a bit like a checksum for applications running in secure Confidential environments.

What do we attest?

- We are communicating with **a genuine and correctly configured secure enclave**
- The **code loaded inside the enclave** is the expected code
- The attestation report is **genuinely signed by the hardware provider**

Note, that the attestation process does not audit the application code itself, but verifies it has not been tampered with.

Verification

When a client attempts to establish communications with the enclave server, the enclave will send a signed ‘quote’ containing information relating to the previously mentioned elements. The client will verify the quote’s signature and then check it matches a set of expected values.

If any of these **checks fail**, an error is produced and the **user will not be able to communicate with an enclave**.

If these checks are **successful**, the user is able to **communicate** with the enclave **securely using attested TLS,** which we will discuss in the next section.

How does it work in BlindAI DRM?

Remote attestation takes place whenever an external server wishes to connect with the enclave server using BlindAI DRM for the first time.

Pre-attestation

On **building** the enclave, a **manifest file** is created containing a hash of the enclave itself (which also acts as a proxy for the application code), its settings and configuration, etc. Clients attempting to connect with the enclave server will need to place a copy of this manifest.toml file, by default at the root of their BlindAI repo.

Attestation workflow

The key steps behind the attestation process that take place when the custodian server first verifies the enclave server before establishing a connection with it are as follows:

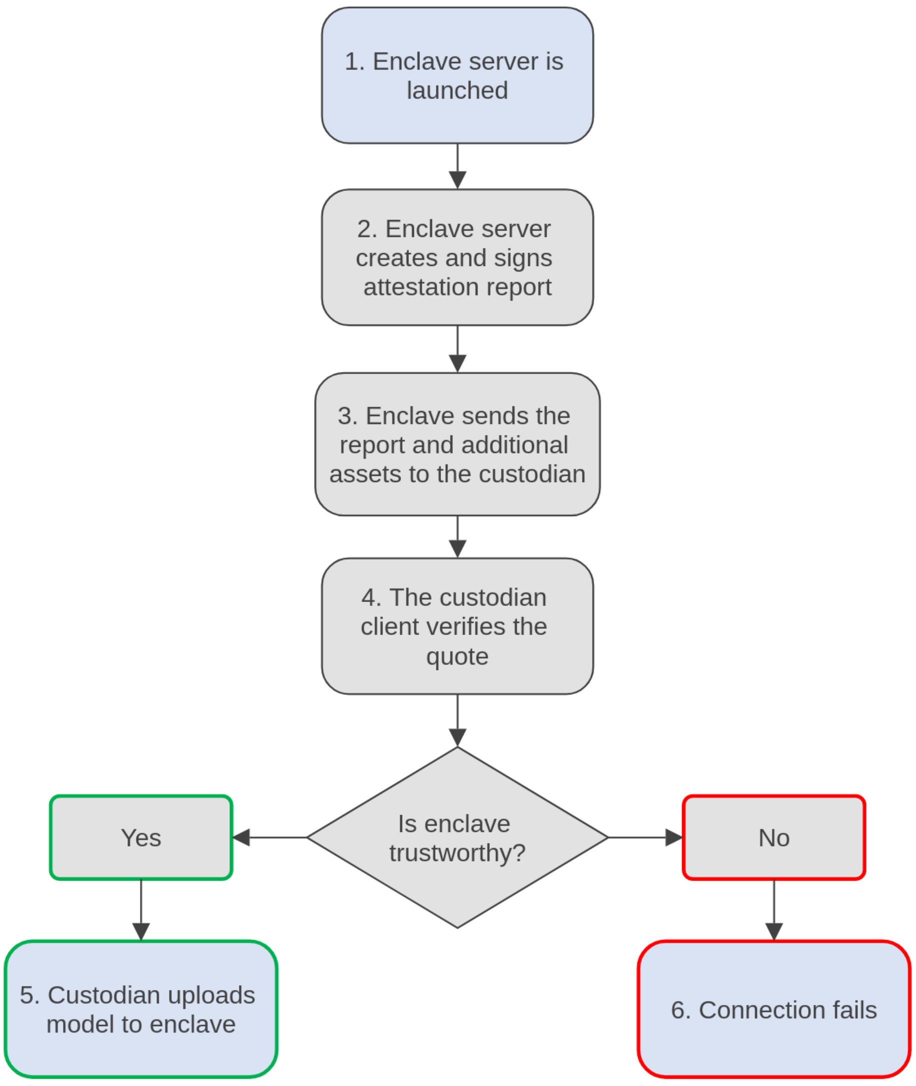
1. Enclave server is launched
2. Enclave server creates and signs attestation report
3. Enclave sends the report and additional assets to the custodian
4. The custodian client verifies the quote
Is enclave trustworthy?
Yes
No
5. Custodian uploads model to enclave
6. Connection fails

Figure 5: Attestation before model upload

1. **Enclave server is launched:** When the enclave server is launched, it will attempt to establish a connection with the custodian server, which will trigger the attestation process.
2. **Enclave server creates and signs attestation report:** The enclave generates an attestation report containing information relating to its contents and configuration. It is signed by the enclave's attestation key, which is certified by a private PCK signing key which chains back to the Intel SGX Root CA. This provides proof that the report has been created by a genuine Intel SGX enclave.
3. **Enclave sends a 'quote' and additional assets to the custodian:** This signed report is placed within a structure called a 'quote' and is sent to the custodian client along with assets referred to as **collateral.** The collateral includes the **PCK certificate** (which includes the public key corresponding to the PCK's private signing key) and a **TLS certificate** so that the client can go on to establish secure communications with the enclave.
4. **The custodian client verifies the quote:**
   a) The PCK certificate is used to verify the attestation key

b) The application code and enclave settings are checked against the client's manifest.toml file to check they are as expected

5. **If the attestation process is successful, the model is uploaded to the enclave:** If the attestation process is successful, the client uses the TLS certificate to establish a secure TLS connection with the enclave and then sends the model to the enclave
6. **If the attestation process fails, the connection fails:** An error is shown to the client and the model is not uploaded

Attested TLS

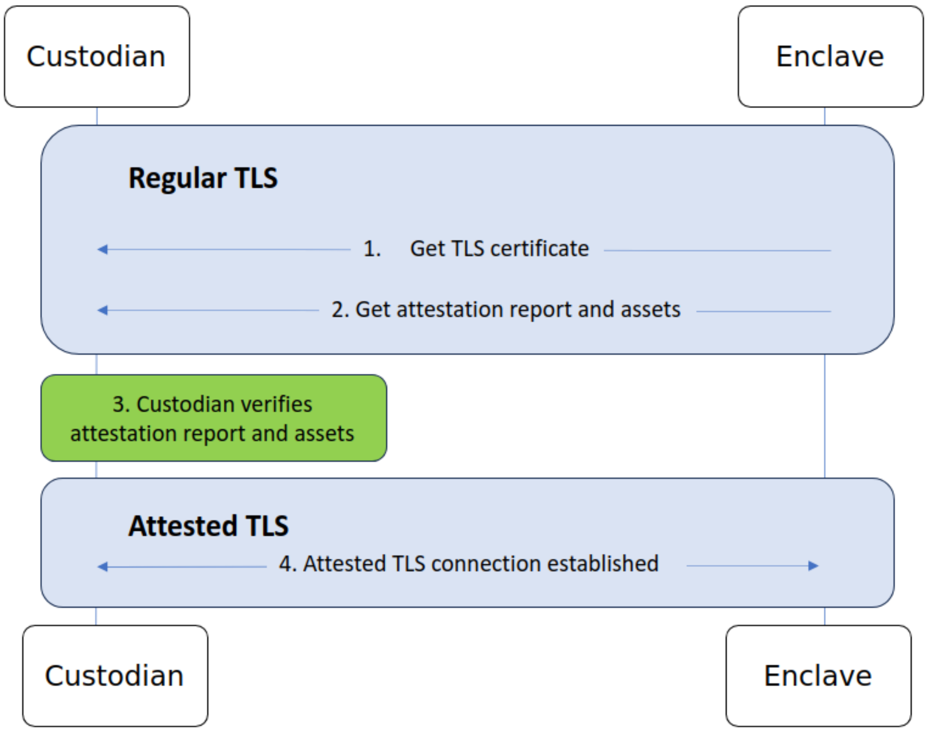
Custodian
Enclave
Regular TLS
1. Get TLS certificate
2. Get attestation report and assets
3. Custodian verifies attestation report and assets
Attested TLS
4. Attested TLS connection established
Custodian
Enclave

Figure 6: Attested TLS workflow

What is attested TLS?

Attested TLS combines the security of data in transit with TLS with a proof of the identity of the server and the code it is running through attestation.

In comparison, regular TLS may protect data in transit robustly, but it does not verify who we are sending it to or what they will do with the data.

By binding a TLS certificate to an attested secure environment we protect ourselves against man-in-the-middle (MITM) attacks, as we have proof that we are communicating with our attested secure environment and not one that is merely forwarding a quote from a valid machine.

How does it work in BlindAI DRM?

The enclave's TLS key pair is generated within the enclave on startup. The enclave's private key is born, lives and dies in the enclave and is therefore never accessible to anyone at any point.

When a client first attempts to establish communications with the enclave server, it will receive a TLS certificate from the server, plus an attestation report which includes a hash of the public key in its attestation report. Since the report is signed by an attestation key that is certified by the PCK key which chains back to the Intel SGX root CA, there exists cryptographic proof that the TLS public key is bound to an intel SGX enclave.

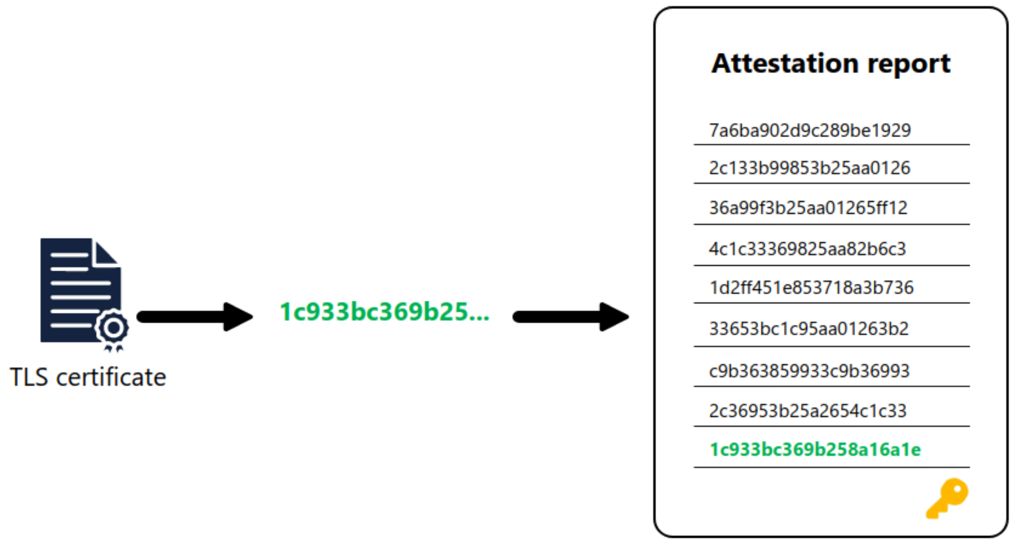

Figure 7. Hashing of TLS in attestation report

Once the enclave's certificate has been verified by a client, secure TLS connection is established with the enclave using the TLS certificate sent by the enclave server to the client.

Introducing BlindAI (original solution)

BlindAI DRM is a variant of our original solution BlindAI that has been adapted for a specific use case and set of objectives.

BlindAI is an open-source solution to deploy and query AI models while guaranteeing data privacy.

BlindAI focuses on two key parties: the AI provider and the end user. We provide a Python client for the end user and a privacy-friendly server for the AI provider.

Models are deployed on the AI provider's infrastructure (with Mithril Security as the AI provider in the case of our API) within secure enclaves. When end users query models deployed with BlindAI, they benefit from robust end-to-end protection, since the AI provider can never access their data due to the enclave's robust isolation. The end user can verify the code integrity and enclave authenticity through attestation.

This solution has been successfully audited by the independent research lab Quarkslab.

Adaptation for BlindAI DRM

With BlindAI DRM, we use the same underlying technologies but the focus shifts. Since we imagine a scenario where an AI borrower serves models on their infrastructure, we focus on protecting AI IP with Confidential technology and adding control features for the custodian so they can limit or revoke access to their AI as they see fit.

We made the following key modifications:

- We added a custodian client enabling the custodian to upload models to the enclave server and attest the enclave.
- We added custodian control features, allowing them to allocate end users a compute budget, track model consumptions and manually shut down the enclave server at any time.

How Confidential technologies help us achieve our goals

We have seen how Confidential technologies help us provide a robust privacy solution for our use case:

Goal	**How we achieve it**
The weights of the model are **never directly accessible** to the AI borrower or consumer	**Secure enclaves** provide robust isolation meaning model weights are never exposed when uploaded to the enclave
Without explicit approval of the custodian, the model is unusable	The weights remain **encrypted** in the enclave and are therefore unusable unless access is granted by the custodian
We offer **code integrity** so the custodian can verify they are talking to a genuine enclave serving the expected code	**Attestation** allows us to verify the enclave and server code

Figure 8: How confidential technologies help us achieve our goals

Conclusions

In this document, we have covered the BlindIA DRM project's key objectives, features, architecture and workflow and detailed the Confidential technologies that help us achieve our goals.

Potential next steps

Confidential Computing can be used to not only have traceable and lockable inference of AI models but can also be applied to training.

It is possible to have a similar DRM mechanism with full visibility for the custodian on what algorithm and training data is used. Instead of having a computational budget that is unlocked in the form of the number of inferences, we could replace it with a number of epochs.

For instance, the custodian could be contacted at boot time, and the AI producer could ask for 10 epochs, consume those, and be blocked until the custodian grants more budget.

This provides for the custodian:

- Full visibility on the amount of compute used throughout the training.
- Control over the compute allowed, with a manual off-switch available to block all usage at any time, and a “dead man’s switch” cutoff of further training if contact with the custodian is lost. The weights always remain encrypted with keys owned only by the custodian and are unusable unless access is granted by the custodian.