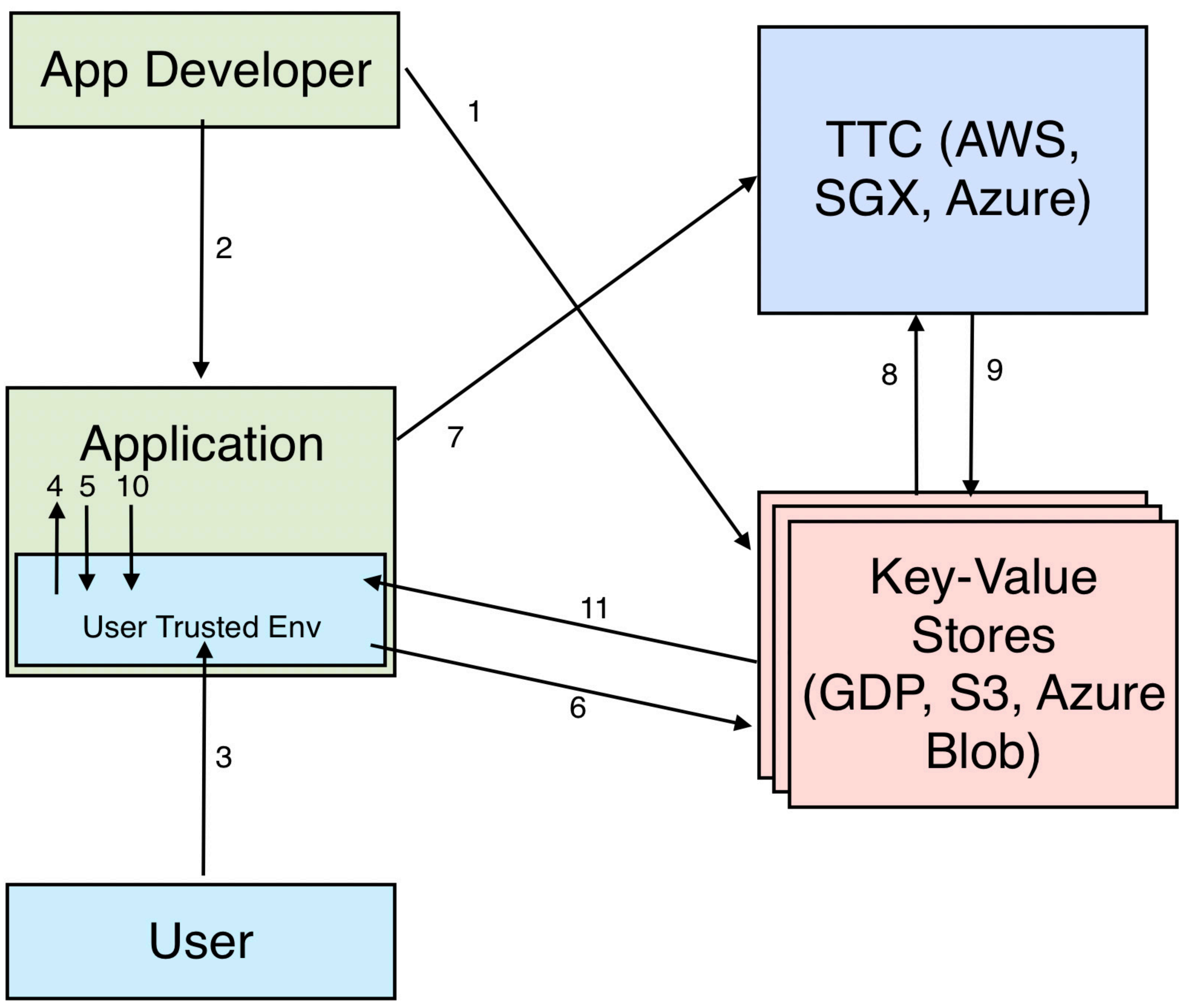
A Privacy Preserving Machine Learning System

Alexander Wu, Hank O’Brien, Joseph Gonzalez, John Kubiatowicz

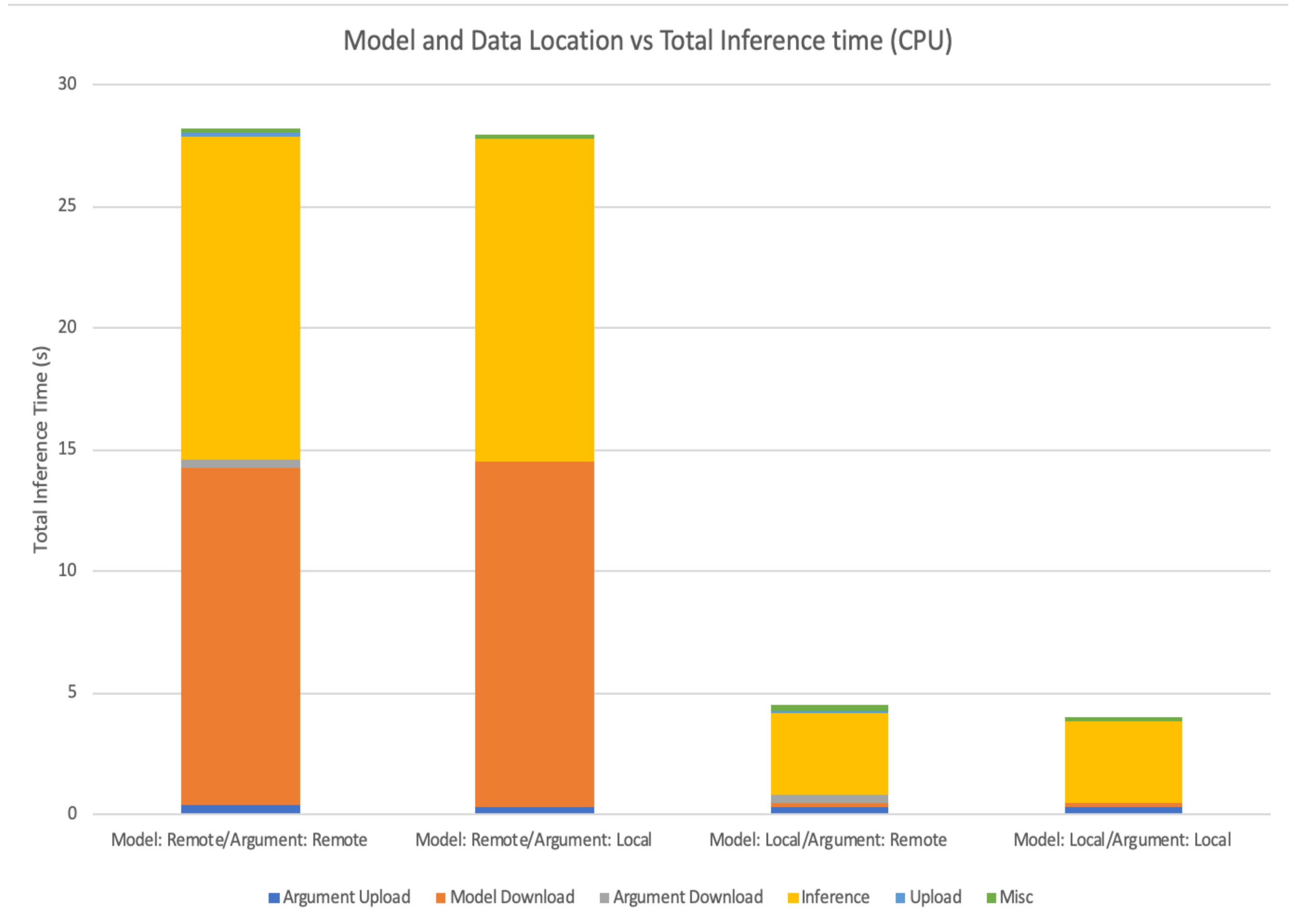
System Overview



Abstract

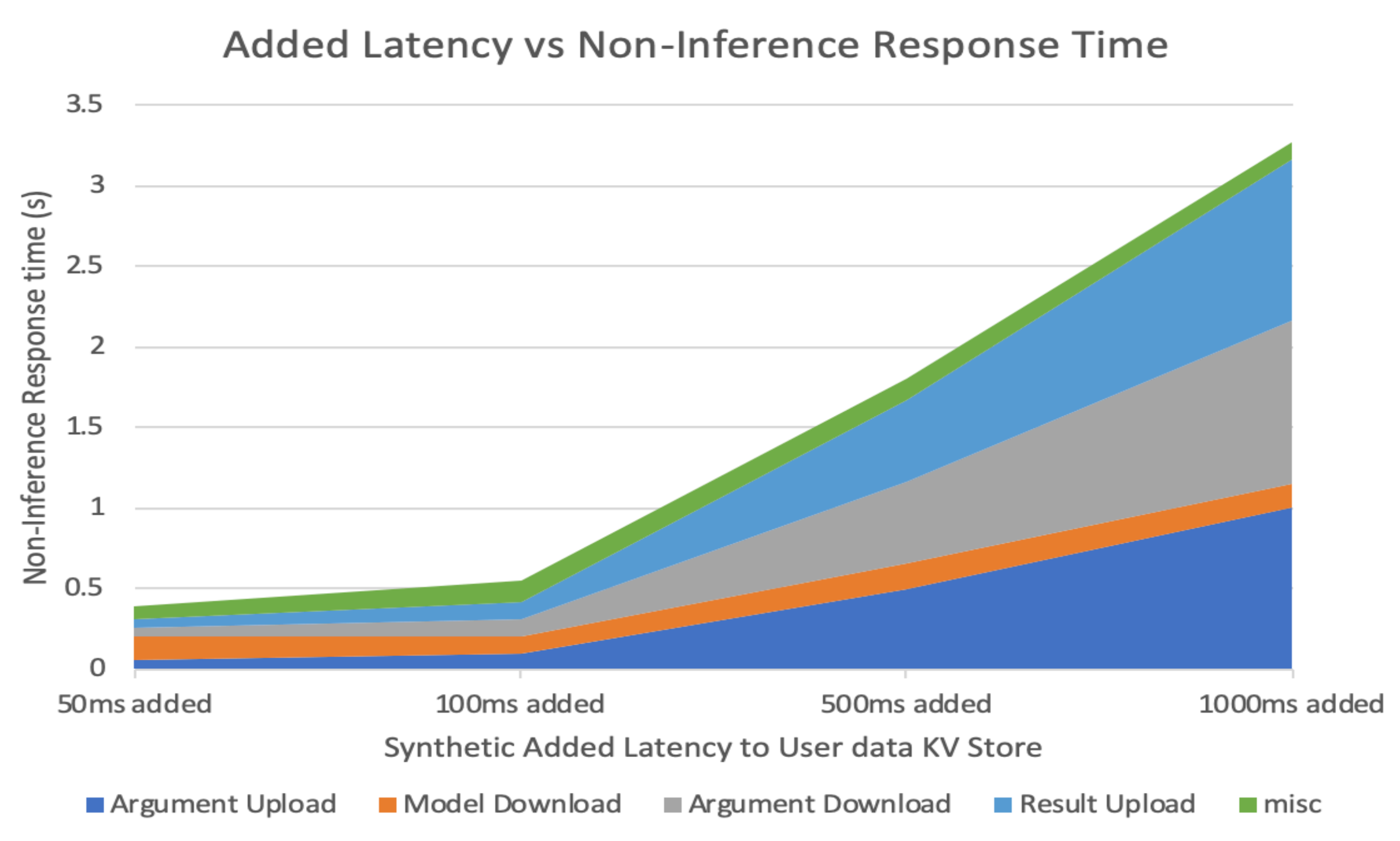
It is generally agreed that machine learning models may contain intellectual property which should not be shared with users, while at the same time there should be mechanisms in place to prevent the abuse of sensitive user data. We propose a machine learning inference system which provides an end-to-end preservation of privacy for both a machine learning model developer and user. Our system aims to minimize its constraints on the expressiveness and accuracy of machine learning models. Our system achieves this by utilizing trusted computation, with a trust-performance tradeoff which extends to a cryptographic proof that data is not tampered with.

Results



Model: Resnet50v2 (90MB compressed)
Input data: a 1.7MB photo
Output Data: ~20B of text
n=40

The first observation here is that we should aim to place the model on a KV Store with as low latency as possible to the TTC. We see this here when the model is on Azure as it must be downloaded to the TTC hosted on computers in RISE before any inference can take place, but because our design is serverless and we have not yet implemented any sort of model-prefetching, this becomes part of the critical path. This effect is actually even more pronounced than these charts suggest; we were unable to get our ONNX runtime to utilize the GPUs available on these machines, but if we were, then we anticipate the inference stage (in yellow) being an order of magnitude faster which would make the orange “model download” contribute in an even more outsized way than it currently does. These tests were all done on a machine with 2, 24-core, 2.6GHz Haswell-generation Intel Xeon processors with 256GB DDR4 RAM and about a 1Gbps network connection (estimated from running speed tests from the machine).



Inference time was omitted from this graph because it was relatively constant at about 3.40 seconds per test and we wanted to highlight the impact of network latency on our approach

Overview

We use the browser’s built in iframe isolation policy to enforce the untrusted app/UTE barrier and we utilize a message passing interface to communicate between our iframes and the untrusted app. The TTC is a separate product from a third-party infrastructure provider and in its most secure form would use secure enclaves to provide privacy

Network Critical Path

1. App Developer creates a new ML model, encrypts* it, and uploads it to one or more KV Stores
   a. the model is encrypted with the keys for a TTC, if the author wishes to support multiple TTCs, multiple copies of the model are uploaded, each encrypted with a different key
2. App Developer creates a new application with an embedded UTE
3. User uploads user data to the UTE
4. UTE returns a DataDescriptor to the App
5. App developer tells the UTE to publish the user data to a particular KV store and to prepare it for a particular TTC
6. Data is encrypted by the UTE and uploaded to the KV store
7. Application initiates a serve request with a model handle and data handle
8. TTC downloads model and handle from (potentially different) KV Stores and decrypts both, and runs the model on this data
9. TTC re-encrypts the inference result and uploads it to a KV store
10. Application receives a response handle and asks UTE to receive the corresponding result
11. UTE downloads and decrypts the inference result and renders it to the user

* Crypto has not yet been fully implemented, but we are working on using the Signal Protocol to establish an asynchronous, end-to-end encrypted channel between the UTE and TTC

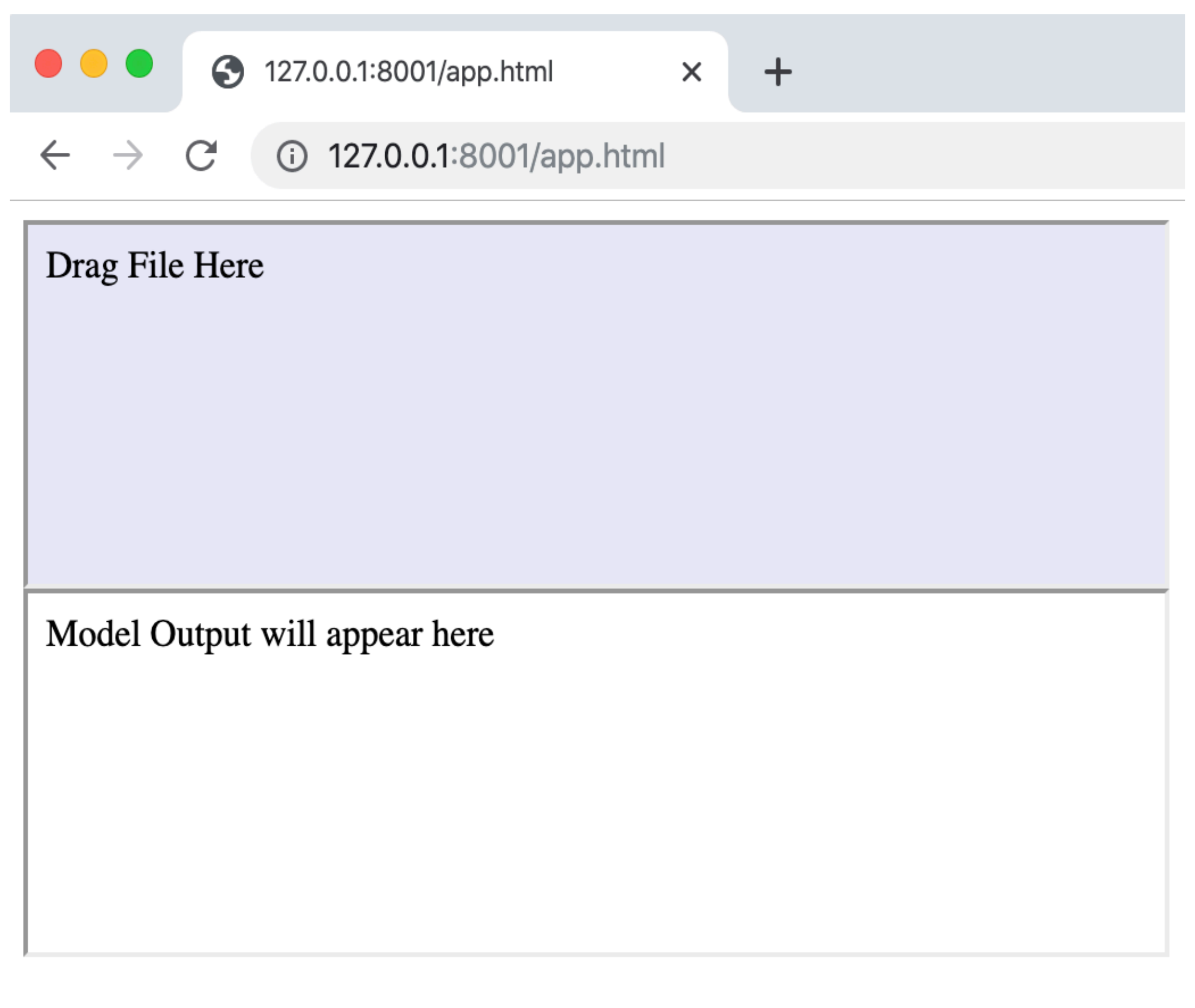
Cryptography

Cryptographic hardening primarily relies upon the Signal Protocol, a well-established asynchronous double ratchet protocol. The protocol establishes a confidential channel between the UTE and TTC. Here, we use the abstraction that each machine is assigned its own identity key. This public component of the key can be signed by a certificate authority in less secure settings. In more secure settings, remote attestation can be used to prove that the key originated from an enclave, and that was generated and secured by a publicly available program. Note that this abstraction does not prevent traditional scaling and load balancing techniques which involve secure key sharing schemes.

Future Work

Our first priority in the near term is finishing the Signal integration and fixing our ONNX issues with GPUs. On a more medium term however, we think adding features like supporting ML models that accept multiple data points and running the ML models in containers with heavily restricted IO permissions to provide further isolation and protection in this scheme. There is also a good deal of long-term future work in two areas, 1) developing a new UTE for desktop and mobile applications, and 2) extending the security of the TTC to support remote attestation and verifiable computing from secure enclave technologies like Intel SGX. Furthermore, there is a good deal more work we could do to further improve overall system latency, such as by initiating the model fetch and the data fetch in parallel when the model is not already on the TTC, or some form of dynamic data replication where the more a model is used in a particular TTC, the more likely it becomes that the data for this model is located nearer this TTC geographically. This will significantly reduce the model fetch time, which is on the inference critical path.

UI Example


127.0.0.1:8001/app.html
127.0.0.1:8001/app.html
Drag File Here
Model Output will appear here