Project Report for
Secure and Privacy-Preserving Deep Learning

The goal of our project is to enable secure, privacy-preserving deep learning through a novel combination of techniques: trusted hardware to provide end-to-end security for computation, differential privacy to provide guaranteed privacy for individuals, and program analysis to determine how to enforce the desired security and privacy properties. The combination will provide strong guarantees of both security and privacy, enabling the collection of enormous amounts of new data for deep learning purposes.

Towards this goal, we have made three primary technical accomplishments, in part supported by the grant. First, we developed the first framework for building private virtual assistants. Second, we designed use control policies to allow individuals to specify protections for their data, data capsules to protect that data, and a static analysis-based enforcement mechanism for computations over data capsules. Third, we developed Helio, which provides high-performance, distributed, secure execution on secure hardware like Intel SGX.

The Private Virtual Assistant

In order to bring the benefits of secure and privacy-preserving computation to users, we have explored the idea of a "private virtual assistant" (PVA) that can help users make decisions and act on the user's behalf. The key idea is to build the PVA as a set of micro-services implemented in composable "applets." Each individual applet is simple, and can be implemented in just a few lines of code. When combined, however, they can perform complex tasks on behalf of the user.

To support this approach, we proposed data capsules, a means of storing encrypted data with a security and privacy policy attached. The data capsule guarantee is that all computations performed on the data in the capsule will adhere to the policy attached. In the PVA setting, each user encodes his or her privacy preferences as use control policies attached to the appropriate data capsules. Applets can use a central index of data capsules to obtain private data of many different types from many different users, provided the applet satisfies all of the relevant policies. The output of each applet is in the form of new data capsules, with residual policies influenced by the behavior of the applet and the policies of its inputs.

To protect data capsules, we presented the concept of use control policies, which specify both who may access the protected data (in the style of existing access control policies), and how they may use the data. The latter kind of restriction allows expressing, for example, that results
computed from the data must be differentially private—a restriction on how the data is to be used, which is impossible to express or enforce in existing approaches.

To enforce data capsule policies, we developed a mechanism based on static analysis of applet implementations. Our applet API, based on Spark's DataFrames, enables representing the applet as a dataflow graph. Analyzing the graph allows determining the security and privacy implications of allowing the applet to run, and computing a residual policy on the data capsule computed by the applet. For example, if an input data capsule has a policy specifying that the “email” field of the input data cannot be used, the dataflow graph can be used to determine whether or not information from that field flows to the output data capsule.

Enforcing more advanced policies—like those requiring differentially private outputs—is more complicated. Existing work on differential privacy has focused on algorithms for particular tasks. Our goal is instead to provide differential privacy in many different applets. Towards this goal, we developed the first approach for providing differential privacy for general-purpose data analytics queries that supports the common constructs used in practice. Our approach uses the applet’s dataflow graph to determine the sensitivity of the applet’s computation, then modifies the graph to add random noise to the results based on this sensitivity. Our dataflow analysis can thus be used to automatically provide advanced security and privacy properties like differential privacy for general-purpose analytics. When the target applets compute statistical results and involve large samples, our approach can provide good utility for a wide variety of computations.

Another advantage of the applet API is the ability to compile applets into programs suitable for secure execution using secure hardware like Intel SGX. SGX provides secure enclaves that execute a single binary; we designed an intermediate representation (IR) language for the operations and expression types in our API, and then implemented a compiler from the IR language to enclave binaries.

**Helio: Distributed Secure & Privacy-preserving Computation**

Our goal for the PVA platform is to provide high-performance distributed execution of applets, with the potential to scale to millions of users. To achieve this goal, we developed Helio, a framework for distributed secure and privacy-preserving computation. Helio leverages secure hardware like Intel SGX to run programs on untrusted infrastructure (for example, in the untrusted cloud), while guaranteeing confidentiality and integrity of the computation results.

Helio works in the same way as distributed data processing frameworks like MapReduce and Apache Spark. Our implementation is based on Spark, but for executing pieces of computation that operate on data capsules, our framework uses Helio's SGX bridge to construct a secure enclave, stream the encrypted data into the enclave, decrypt it, run the computation, and encrypt the results. The Helio framework thus scales with the size of the compute cluster used, like MapReduce and Spark, but guarantees confidentiality and integrity of the results.
Conclusions

Towards our final goal of enabling large-scale decision-making based on sensitive data, we have made progress in three areas. While these projects are still in their research prototype stage, the exploration of the areas have been fruitful. First, as part of the PVA project, we have explored the use of user-specified security and privacy policies, proposing the concept of data capsules and use control policies. Second, we have begun exploring the use of static analysis of PVA applets to enforce policies on data capsules, including those with advanced privacy requirements like differential privacy. Third, in the Helio project, we have been developing a scalable framework for distributed secure computation, guaranteeing both the confidentiality and integrity of results.

Together, the three represent an initial step towards an end-to-end guarantee: the user specifies a security and privacy policy that can include requirements like differential privacy, and the system guarantees that all computed results respect this policy. Our initial prototypes demonstrate that this goal is feasible. Results from the project has led to papers under submission, and code for the prototype implementation of our PVA framework has been contributed to the BDD repo.