A verification framework for secure machine learning

December 12, 2019

Advisors: Karthikeyan Bhargavan and Prasad Naldurg
<firstname.lastname@inria.fr>

Institute: PROSECCO, Inria Paris

Address: 2 Rue Simone Iff, 75012, Paris, France

Language: English

Skills with a strong desire to learn

- basic knowledge of machine learning techniques and algorithms, data modeling and evaluation, experience with machine learning libraries and packages
- cryptography, cryptographic primitives including homomorphic encryption, secure multiparty computation, garbled circuits, implementations
- programming languages and security, security protocol modeling, functional programming, information flow analysis, type systems, formal verification
- security and privacy properties, differential privacy, confidentiality

Introduction

Machine learning applications consume vast amounts of user data (including PII data) to produce analytical models, which are subsequently used to assist in making classification decisions on new data without human intervention. While machine learning applications are getting more and more sophisticated, security and privacy issues in this context have received lesser attention. In privacy preserving machine learning (PPML), such a machine-learning classifier (server) should treat user queries opaquely, and should not learn anything about the query issued by a client or its resulting response (i.e., the resulting class). A client should only learn the correct response to its query and not learn anything about the model parameters on the servers. Achieving these goals, simple as they sound, turns out to be quite difficult and expensive in practice, and is an active area of research.

Our goal in this project is to explore the landscape of secure machine learning, including PPML, and build implementations of state of the art verifiable protocols and solutions that are also fast and efficient. There are many layers in this quest:

1. At a high level, secure machine learning involves the specification of a set of desirable security and privacy properties, from the viewpoint of both clients (end users of the machine learning service) and servers, who have the analytical models. Since client data is used to produce these models, in addition to the classifier properties highlighted above, the notion of protecting PII (personally identifiable information) during learning is also important. Once these properties are specified, protocols to achieve these properties for different machine learning techniques need to be specified and analyzed for their compliance. At this level, the lower level cryptographic techniques that are required to meet the security goals are specified abstractly, similar to the Dolev-Yao symbolic models and we need novel programming language and verification techniques to track and validate information flow properties, in addition to validating functional correctness of machine learning goals, across multiple rounds of interaction between the clients and servers.
2. These protocols in turn rely on novel and sophisticated cryptographic algorithms, such as homomorphic encryption (HE) [5, 4, 7, 6] where computations can be performed directly on encrypted data to give an encrypted result, secure multi-party computation (SMC) [10, 8], garbled circuits [6, 7], and functional encryption (FE) [3]. All of these techniques, specifically HE, are currently impractical for machine learning. Clever combinations of partial HE and SMC or GC, interleaved with randomization, masking, and permutations as shown in these works, attempt to bridge the gap between security and performance. Through our research in building verified implementations of cryptographic primitives, protocols and applications [2] we observe that they can be difficult to implement correctly, and errors and vulnerabilities in their code can remain undiscovered for

long periods before they are exploited. Similar to our work on HACL$*$ we plan to explore how to build verified implementations of these new primitives, which will necessarily have different verification goals and functional correctness specifications from the primitives we have studied earlier. In fact, the choice of the cryptographic solution will be based on the security guarantees they can offer in the client-server machine learning setting, as well as a careful analysis of their performance overheads.

3. The lowest layer of this framework is the verified implementation of the client and server code. This code is derived from the verified implementations of the earlier layers and translated into C code using effective compiler tools. The low-level implementations preserve the broader security goals from the higher layers within the machine learning context, as well as provide memory safety, functional correctness, and resilience to some forms of side channel attacks. This last layer provides us the performance guarantees, making verified low-level code for secure machine learning as performant as the fastest implementations of these techniques This generated code can be dropped as modules into existing machine libraries and programming frameworks, for example replacing native computations by their homomorphically secure computations to assert confidentiality properties intuitively. Additionally, we can compile the whole protocol into separate client and server modules to form an end-to-end verified solution that can run over an insecure network, with these modules containing all the functionality as well as security code, and provide strong guarantees, without sacrificing performance.

The glue that holds all these layers is the F$*$. F* is an ML-like functional programming language with a type system that includes polymorphism, dependent types, monadic effects, refinement types, and a weakest precondition calculus [1]. The language is aimed at program verification, and its type system allows the expression of precise and compact functional correctness and security property specifications for programs, which can be mechanically verified, with the help of an SMT solver. After verification, and F* program can be compiled to OCaml, F#, C, or even WebAssembly, and so it can run in a variety of platforms.

With our initial experience in building a protocol to do secure classification, we show in [9], that even the act of comparing two numbers homomorphically, can reveal information that can be exploited to learn features of an analytical models unintentionally. We believe that this is a rich area for research, and there are many open problems, including foundational and definitional ones in understanding the nature and scope of security and privacy guarantees for machine learning and their verification. We believe that these efforts need to be enmeshed with innovations in machine learning techniques and algorithms and can only be realized through a concerted and integrated effort.

As a part of this ongoing effort, we seek brilliant students who are interested in machine learning and security, and are willing to explore verification tools and programming languages techniques to build verified solutions in what we believe is a fundamental requirement in this area of secure machine learning, something that has been often neglected or deemed impractical for performance reasons. We want to view the approach outlined here as a guideline to pick interesting sub projects in the field, and are open to suggestions including ideas on differential privacy and the use of secure enclaves, as well as new insights and problems in this framework. We seek both Ph. D students and postdoctoral students, and students will be expected to contribute to a clearly defined sub-task, with the goal of leading to a dossier of publications in top research conferences and contribute to foundational aspects of this field.

References

[1] F*: A higher-order effectful language designed for program verification. `https://www.fstar-lang.org/`. Accessed: 2018-11-01.

[2] Karthikeyan Bhargavan and Prasad Naldurg. Practical Formal Methods for Real World Cryptography (Invited Talk). In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019)*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:12, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[3] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference (TCC)*, pages 253–273, 2011.

[4] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[5] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[6] Trinabh Gupta, Henrique Fingler, Lorenzo Alvisi, and Michael Walfish. Pretzel: Email encryption and provider-supplied functions are compatible. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 169–182, 2017.

[7] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, Baltimore, MD, 2018. USENIX Association.

[8] Eleftheria Makri, Dragos Rotaru, Nigel P. Smart, and Frederik Vercauteren. Epic: Efficient private image classification (or: Learning from the masters). Cryptology ePrint Archive, Report 2017/1190, 2017. `https://eprint.iacr.org/2017/1190`.

[9] Prasad Naldurg and Karthikeyan Bhargavan. A verification framework for secure machine learning. PPML '19: Privacy Preserving Machine Learning, Workshop with ACM CCS '19 London, 2019.

[10] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: Efficient and private neural network training. In *Privacy Enhancing Technologies Symposium*. (PETS 2019), 2019.