# Efficient Formally Secure Compilers to a Tagged Architecture

## General Information

**Advisor:** Cătălin Hriţcu ⟨catalin.hritcu@gmail.com⟩

**Institition:** INRIA Paris, Prosecco Team

**Location:** 2 rue Simone Iff, 75012 Paris, France

**Language:** English

**Existing skills or strong desire to learn:**

- compilation (for imperative or functional languages);
- functional programming (e.g., ML or Haskell);
- security, full abstraction, gradual typing;
- optional: formal verification in Coq or F$^\star$ [4];

## Context

Severe low-level vulnerabilities abound in today's computer systems, allowing cyber-attackers to remotely gain full control. This happens in big part because our programming languages, compilers, and architectures were designed in an era of scarce hardware resources and too often trade off security for efficiency. The semantics of mainstream low-level languages like C is inherently insecure, and even for safer languages, establishing security with respect to a high-level semantics does not guarantee the absence of low-level attacks. *Secure compilation* using the coarse-grained protection mechanisms provided by mainstream hardware architectures would be too inefficient for most practical scenarios. This long-term project is aimed at leveraging emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilers for realistic programming languages, both low-level (the C language) and high-level (ML and a dependently-typed variant). These compilers will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilers will target a *tagged architecture* [1, 2], which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We will experimentally evaluate and carefully optimize the efficiency of our secure compilers on realistic workloads and standard benchmark suites. We will use property-based testing and formal verification to provide high confidence that our compilers are indeed secure. Formally, we will construct machine-checked proofs of *full abstraction* with respect to a secure high-level semantics [3]. This is much stronger than just compiler correctness and ensures that no machine-code attacker can do more harm to securely compiled components than a component in the secure source language already could.

## Topics

We are looking for talented, highly motivated students and young researchers to work with us on this project. This can happen in the context of research internships or PhD, PostDoc, Starting Researcher positions. Potential topics include, but are not restricted to:

- protecting mutually distrustful, low-level components [3]
- memory safety for C
- micro-policy composition
- extending C and ML with micro-policies
- secure compilation for ML
- secure compilation (i.e., ML extraction) for F$^\star$ [4]

For more details please get in contact using information above.

## References

[1] A. Azevedo de Amorim, M. Dénès, N. Giannarakis, C. Hriţcu, B. C. Pierce, A. Spector-Zabusky, and A. Tolmach. Micro-policies: Formally verified, tag-based security monitors. *Oakland S&P*. 2015.

[2] U. Dhawan, C. Hriţcu, R. Rubin, N. Vasilakis, S. Chiricescu, J. M. Smith, T. F. Knight, Jr., B. C. Pierce, and A. DeHon. Architectural support for software-defined metadata processing. *ASPLOS*. 2015.

[3] Y. Juglaret, C. Hriţcu, A. Azevedo de Amorim, B. C. Pierce, A. Spector-Zabusky, and A. Tolmach. Towards a fully abstract compiler using micro-policies: Secure compilation for mutually distrustful components. Technical Report, arXiv:1510.00697, 2015.

[4] N. Swamy, C. Hriţcu, C. Keller, A. Rastogi, A. Delignat-Lavaud, S. Forest, K. Bhargavan, C. Fournet, P.-Y. Strub, M. Kohlweiss, J.-K. Zinzindohoue, and S. Zanella-Béguelin. Dependent types and multi-monadic effects in F*. *POPL*, 2016.