Internships on the F* proof-oriented programming language

Advisor: Catalin Hritcu, MPI-SP, Bochum, Germany

November 20, 2025

Introduction

Our group was involved in the design and evolution of F*, a proof assistant and general-purpose functional programming language with effects targeted at program verification. The F* design combines SMT-based automation with the power and expressiveness of proof assistants like Coq, which enables users to prove arbitrarily complex properties using a mix of automatic and interactive proofs [18]. To achieve this we have introduced full dependent types and tracking of side-effects, while isolating a core language of pure total functions that can be used to write specifications and proof terms [23]. A key insight we had in F* is that verification conditions can be computed generically for any effect using *Dijkstra monads* [3, 4, 16, 20, 27], which are computational monads indexed by specification monads. Moreover, carefully exposing the computational monad representation of effects can be used to verify relational properties that characterize many useful notions of security, program refinement, and equivalence [13, 17]. This general treatment of side-effects also allowed us to implement support for meta-programming and tactics in F* simply as a user defined effect [18].

Recently, our group has used these innovations of F* for building a formally secure compilation framework for effectful F* programs using a combination of reference monitoring and higher-order contracts [6, 7, 8]. Our compiler satisfies Robust Relational Hyperproperty Preservation, a very strong secure compilation criterion that is stronger than full abstraction [1]. We also used ideas we initially developed with the aim of relational verification in F* [17], to build SSProve, a foundational framework for modular cryptographic proofs in Coq [2, 14], which won a Distinguished Paper Award at CSF'21 [2]. Closer to practice, F* was used by our collaborators including Inria and Microsoft Research to verify cryptographic libraries [5, 21, 29] that were deployed in Mozilla Firefox and the Linux kernel, as well as secure parsers that were deployed in the Windows kernel [5, 22, 25].

The remainder of this document presents several topics that could be interesting to interns.

1 Dijkstra monads and incorrectness logic

Dijkstra monads are the way F* verifies the *correctness* of programs for arbitrary monadic effects, i.e., it proves the absence of bugs with respect to a specification expressed itself as a monad [3, 16, 27]. More recently the programming languages and verification community has also become interested in proving *incorrectness* [19, 28], i.e., proving the presence of bugs with respect to a specification. We are currently investigating how Dijkstra monads can be dualized from correctness to incorrectness. One challenge is finding dual orderings for correctness and incorrectness specifications, for instance for weakest preconditions.

2 Verification framework for cryptographic security in F*

While some of F*'s biggest successes have involved verifying cryptographic code [5, 11, 21, 29], the formally verified properties have so far been limited to functional correctness and memory safety [20]. The cryp-

tographic security proofs for this code have so far been only done informally on paper, which does not provide enough assurance, especially for implementations of complex protocols such as TLS and QUIC [5]. We would like to solve this issue by building a novel verification framework for formalizing cryptographic security proofs in F*. A first ingredient for this will be a Dijkstra monad for automatically verifying probabilistic properties of cryptographic code, generalizing prior unary program logics for reasoning about the probabilities of bad events [9]. A second ingredient will be a probabilistic relational program logic that is modular in the underlying side-effects [17]. A third ingredient are the higher-level relational reasoning principles of *state-separating proofs (SSP)* [10], which we recently formalized in SSProve, a Coq framework for modular cryptographic proofs [2, 14]. A final ingredient could be using parametricity for providing simple formal proofs of noninterference properties such as side-channel resistance, which F* cryptographic libraries achieve by careful type abstraction [20, 29], but which still remains to be carefully formalized.

3 F* foundations: demystifying the ghost and divergence effects

Over the past years, we have demystified many of the features of F*, such as Dijkstra Monads [16, 27] and Monotonic State [4, 7], by showing how they can (at least in theory) be defined on top of standard dependent type theories. Two of the primitive effects of F* have so not yet been illuminated in this way though: the ghost and divergence effects. In practice, the fact that these effects currently do not have a Dijkstra Monad representation prevents extrinsic reasoning by monadic reification about code with these side-effects. The ghost effect that allows for computationally irrelevant code could maybe be given a representation in terms of a free monad with specific operations [7]. Also the connection between F*'s ghost effect and Extensional Type Theory could be interesting to investigate [26].

The open problem of providing a Dijkstra Monad representation for Div and the other F* effects that build on this effect seems much more challenging. When computing the universe of arrow types with such effects F* discards the universe of the effect representation and also the universe of the result type. This is used in particular to implement in F* divergent state effects that allow for higher-order state [24]. It could be that ideas from guarded type theory [15] or guarded interaction trees [12] could help build intuition for this.

Finally, the relation between these effects and subtyping is also somewhat unclear, and in fact soundness problems were recently found when allowing subtyping for ghost or divergent arrows.[1]

References

[1] C. Abate, R. Blanco, D. Garg, C. Hritcu, M. Patrignani, and J. Thibault. Journey beyond full abstraction: Exploring robust property preservation for secure compilation. *CSF*. 2019.

[2] C. Abate, P. G. Haselwarter, E. Rivas, A. V. Muylder, T. Winterhalter, C. Hriţcu, K. Maillard, and B. Spitters. SSProve: A foundational framework for modular cryptographic proofs in Coq. *CSF*. 2021.

[3] D. Ahman, C. Hriţcu, K. Maillard, G. Martínez, G. Plotkin, J. Protzenko, A. Rastogi, and N. Swamy. Dijkstra monads for free. *POPL*. 2017.

[4] D. Ahman, C. Fournet, C. Hriţcu, K. Maillard, A. Rastogi, and N. Swamy. Recalling a witness: Foundations and applications of monotonic state. *PACMPL*, 2(POPL):65:1–65:30, 2018.

[5] D. Ahman, K. Bhargavan, B. Bond, J. Bosamiya, C. Brzuska, A. Delignat-Lavaud, C. Fournet, A. Fromherz, S. Gibson, C. Hawblitzel, C. Hriţcu, M. Kohlweiss, G. Martínez, H. Ni, B. Parno, J. Protzenko, T. Ramananandro, A. Rastogi, E. Rivas, N. Swamy, and S. Zanella-Béguelin. Project everest: Perspectives from developing industrial-grade high-assurance software. 2025.

[6] C.-C. Andrici, Ştefan Ciobâcă, C. Hriţcu, G. Martínez, E. Rivas, Éric Tanter, and T. Winterhalter. Securing verified IO programs against unverified code in F*. *PACMPL*, 8(POPL):74:1–74:34, 2024.

[1] https://github.com/FStarLang/FStar/issues/3659 and https://github.com/FStarLang/FStar/issues/3644

[7] C.-C. Andrici, D. Ahman, C. Hriţcu, R. Icleanu, G. Martínez, E. Rivas, and T. Winterhalter. SecRef*: Securely sharing mutable references between verified and unverified code in F*. arXiv:2503.00404; to appear at ICFP'25, 2025.

[8] C.-C. Andrici, D. Ahman, C. Hriţcu, G. Martínez, A. Pribisova, E. Rivas, and T. Winterhalter. Towards formally secure compilation of verified F* programs against unverified ml contexts (extended abstract). Draft, 2025.

[9] G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub. A program logic for union bounds. In I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016.

[10] C. Brzuska, A. Delignat-Lavaud, C. Fournet, K. Kohbrok, and M. Kohlweiss. State separation for code-based game-playing proofs. In *ASIACRYPT*. 2018.

[11] A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, J. Protzenko, A. Rastogi, N. Swamy, S. Z. Béguelin, K. Bhargavan, J. Pan, and J. K. Zinzindohoue. Implementing and proving the TLS 1.3 record layer. *IEEE S&P*. 2017.

[12] D. Frumin, A. Timany, and L. Birkedal. Modular denotational semantics for effects with guarded interaction trees. *Proc. ACM Program. Lang.*, 8(POPL):332–361, 2024.

[13] N. Grimm, K. Maillard, C. Fournet, C. Hriţcu, M. Maffei, J. Protzenko, T. Ramananandro, A. Rastogi, N. Swamy, and S. Zanella-Béguelin. A monadic framework for relational verification: Applied to information security, program equivalence, and optimizations. *CPP*, 2018.

[14] P. G. Haselwarter, E. Rivas, A. V. Muylder, T. Winterhalter, C. Abate, N. Sidorenco, C. Hriţcu, K. Maillard, and B. Spitters. SSProve: A foundational framework for modular cryptographic proofs in Coq. *TOPLAS*, 45(3), 2023.

[15] R. Jung, R. Krebbers, J. Jourdan, A. Bizjak, L. Birkedal, and D. Dreyer. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *J. Funct. Program.*, 28:e20, 2018.

[16] K. Maillard, D. Ahman, R. Atkey, G. Martínez, C. Hriţcu, E. Rivas, and E. Tanter. Dijkstra monads for all. *PACMPL*, 3(ICFP), 2019.

[17] K. Maillard, C. Hriţcu, E. Rivas, and A. Van Muylder. The next 700 relational program logics. *PACMPL*, 4(POPL):4:1–4:33, 2020.

[18] G. Martínez, D. Ahman, V. Dumitrescu, N. Giannarakis, C. Hawblitzel, C. Hriţcu, M. Narasimhamurthy, Z. Paraskevopoulou, C. Pit-Claudel, J. Protzenko, T. Ramananandro, A. Rastogi, and N. Swamy. Meta-F*: Proof automation with SMT, tactics, and metaprograms. *ESOP*. 2019.

[19] P. W. O'Hearn. Incorrectness logic. *Proc. ACM Program. Lang.*, 4(POPL):10:1–10:32, 2020.

[20] J. Protzenko, J.-K. Zinzindohoué, A. Rastogi, T. Ramananandro, P. Wang, S. Zanella-Béguelin, A. Delignat-Lavaud, C. Hriţcu, K. Bhargavan, C. Fournet, and N. Swamy. Verified low-level programming embedded in F*. *PACMPL*, 1(ICFP):17:1–17:29, 2017.

[21] J. Protzenko, B. Parno, A. Fromherz, C. Hawblitzel, M. Polubelova, K. Bhargavan, B. Beurdouche, J. Choi, A. Delignat-Lavaud, C. Fournet, N. Kulatova, T. Ramananandro, A. Rastogi, N. Swamy, C. M. Wintersteiger, and S. Z. Béguelin. EverCrypt: A fast, verified, cross-platform cryptographic provider. *IEEE S&P*. 2020.

[22] T. Ramananandro, A. Delignat-Lavaud, C. Fournet, N. Swamy, T. Chajed, N. Kobeissi, and J. Protzenko. EverParse: Verified secure zero-copy parsers for authenticated message formats. *USENIX Security*. 2019.

[23] 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.

[24] N. Swamy, C. Hriţcu, C. Keller, A. Rastogi, A. Delignat-Lavaud, S. Forest, K. Bhargavan, C. Fournet, P.-Y.

Strub, M. Kohlweiss, J.-K. Zinzindohoué, and S. Zanella-Béguelin. Dependent types and multi-monadic effects in F*. *POPL*. 2016.

[25] N. Swamy, T. Ramananandro, A. Rastogi, I. Spiridonova, H. Ni, D. Malloy, J. Vazquez, M. Tang, O. Cardona, and A. Gupta. Hardening attack surfaces with formally proven binary format parsers. In R. Jhala and I. Dillig, editors, *PLDI '22: 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, San Diego, CA, USA, June 13 - 17, 2022*. 2022.

[26] T. Winterhalter. Dependent ghosts have a reflection for free. *Proc. ACM Program. Lang.*, 8(ICFP): 630–658, 2024.

[27] T. Winterhalter, C.-C. Andrici, C. Hriţcu, K. Maillard, G. Martínez, and E. Rivas. Partial Dijkstra monads for all. Extended abstract of presentation at the 28th International Conference on Types for Proofs and Programs (TYPES), 2022.

[28] N. Zilberstein, D. Dreyer, and A. Silva. Outcome logic: A unifying foundation for correctness and incorrectness reasoning. *Proc. ACM Program. Lang.*, 7(OOPSLA1):522–550, 2023.

[29] J.-K. Zinzindohoué, K. Bhargavan, J. Protzenko, and B. Beurdouche. HACL*: A verified modern cryptographic library. *CCS*, 2017.