Activity Report 2019

Project-Team PROSECCO

Programming securely with cryptography

RESEARCH CENTER
Paris

THEME
Security and Confidentiality

Table of contents

1. Team, Visitors, External Collaborators **1**
2. Overall Objectives **2**
2.1.1. New programming languages for verified software 3
2.1.2. Symbolic verification of cryptographic applications 3
2.1.3. Computational verification of cryptographic applications 3
2.1.4. Efficient formally secure compilers for tagged architectures 4
2.1.5. Building provably secure web applications 4
3. Research Program **4**
3.1. Symbolic verification of cryptographic applications 4
3.1.1. Verifying cryptographic protocols with ProVerif 4
3.1.2. Verifying security APIs using Tookan 5
3.1.3. Verifying cryptographic applications using F* 5
3.2. Computational verification of cryptographic applications 6
3.3. F*: A Higher-Order Effectful Language for Program Verification 6
3.4. Efficient Formally Secure Compilers to a Tagged Architecture 6
3.5. Provably secure web applications 7
3.6. Design and Verification of next-generation protocols: identity, blockchains, and messaging 8
4. Application Domains **8**
4.1. Cryptographic Protocol Libraries 8
4.2. Hardware-based security APIs 8
4.3. Web application security 8
5. Highlights of the Year **9**
6. New Software and Platforms **9**
6.1. Cryptosense Analyzer 9
6.2. CryptoVerif 10
6.3. F* 11
6.4. miTLS 11
6.5. ProVerif 11
6.6. HACL* 12
7. New Results **12**
7.1. Verification of security protocols 12
7.2. Verified Software for Cryptographic Web Applications 13
7.3. Journey beyond full abstraction 13
7.4. Principles of Program Verification for Arbitrary Monadic Effects 13
7.5. Meta-F*: Proof automation with SMT, Tactics, and Metaprograms 14
8. Bilateral Contracts and Grants with Industry **14**
9. Partnerships and Cooperations **14**
9.1. National Initiatives 14
9.1.1.1. AnaStaSec 14
9.1.1.2. AJACS 15
9.1.1.3. SafeTLS 15
9.1.1.4. TECAP 15
9.2. European Initiatives 16
9.2.1.1. ERC Consolidator Grant: CIRCUS 16
9.2.1.2. ERC Starting Grant: SECOMP 16
9.2.1.3. NEXTLEAP (304) 16
9.3. International Initiatives 17
9.3.1. Inria International Partners 17
9.3.2. Participation in Other International Programs 17

9.3.2.1. SSITH/HOPE 17
9.3.2.2. Everest Expedition 18
9.4. International Research Visitors 18
9.4.1. Visits of International Scientists 18
9.4.2. Visits to International Teams 19
10. Dissemination **19**
10.1. Promoting Scientific Activities 19
10.1.1. Scientific Events: Organisation 19
10.1.1.1. General Chair, Scientific Chair 19
10.1.1.2. Member of the Organizing Committees 19
10.1.2. Scientific Events: Selection 20
10.1.2.1. Chair of Conference Program Committees 20
10.1.2.2. Member of the Conference Program Committees 20
10.1.3. Journal 20
10.1.4. Invited Talks 20
10.1.5. Leadership within the Scientific Community 20
10.1.6. Scientific Expertise 20
10.1.7. Research Administration 20
10.2. Teaching - Supervision - Juries 20
10.2.1. Teaching 20
10.2.2. Supervision 21
10.2.3. Juries 21
10.3. Popularization 21
11. Bibliography **21**

Project-Team PROSECCO

Creation of the Team: 2012 January 01, updated into Project-Team: 2012 July 01

Keywords:

Computer Science and Digital Science:

A1.1. - Architectures
A1.1.8. - Security of architectures
A1.2. - Networks
A1.2.8. - Network security
A1.3. - Distributed Systems
A2. - Software
A2.1. - Programming Languages
A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.7. - Distributed programming
A2.1.11. - Proof languages
A2.2. - Compilation
A2.2.1. - Static analysis
A2.2.5. - Run-time systems
A2.4. - Formal method for verification, reliability, certification
A2.4.2. - Model-checking
A2.4.3. - Proofs
A2.5. - Software engineering
A4. - Security and privacy
A4.3. - Cryptography
A4.3.3. - Cryptographic protocols
A4.5. - Formal methods for security
A4.6. - Authentication
A4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

B6. - IT and telecom
B6.1. - Software industry
B6.1.1. - Software engineering
B6.3. - Network functions
B6.3.1. - Web
B6.3.2. - Network protocols
B6.4. - Internet of things
B9. - Society and Knowledge
B9.10. - Privacy

1. Team, Visitors, External Collaborators

Research Scientists

Karthikeyan Bhargavan [Team leader, Inria, Senior Researcher, HDR]
Bruno Blanchet [Inria, Senior Researcher, HDR]
Harry Halpin [Inria, Advanced Research Position, until Aug 2019]
Catalin Hritcu [Inria, Researcher, HDR]
Prasad Naldurg [Inria, Advanced Research Position]
Exequiel Rivas Gadda [Inria, Starting Research Position, from Mar 2019]
Eric Tanter [Inria, Advanced Research Position]

Post-Doctoral Fellow

Roberto Blanco Martinez [Inria, Post-Doctoral Fellow]

PhD Students

Carmine Abate [Inria, PhD Student]
Benjamin Beurdouche [Inria, PhD Student, granted by FP7 ERC CIRCUS project]
Natalia Kulatova [Inria, PhD Student]
Benjamin Lipp [Inria, PhD Student]
Kenji Maillard [Ecole Normale Supérieure Paris, PhD Student, until Nov 2019]
Denis Merigoux [Inria, PhD Student]
Marina Polubelova [Inria, PhD Student]
Jérémy Thibault [Inria, PhD Student]

Technical staff

Cezar Constantin Andrici [Inria, Engineer, from Aug 2019 until Nov 2019]
Iness Ben Guirat [Inria, Engineer, until Apr 2019]
Adrien Durier [Inria, Engineer, from Sep 2019]
Florian Groult [Inria, Engineer]
Theo Laurent [Inria, Engineer]
Ramkumar Ramachandra [Inria, Engineer, from Aug 2019]
Antoine Van Muylder [Inria, Engineer, from Nov 2019]

Interns and Apprentices

Guillaume Gette [Inria, from Apr 2019 until Sep 2019]
Elisabeth Labrada Deniz [Inria, until Jan 2019]
Antoine Van Muylder [Inria, from May 2019 until Sep 2019]
Mikhail Volkhov [Inria, from Apr 2019 until Aug 2019]

Administrative Assistants

Christelle Guiziou [Inria, Administrative Assistant]
Mathieu Mourey [Inria, Administrative Assistant]

External Collaborators

Victor Dumitrescu [Nomadic Labs]
Guido Martinez [CIFASIS-CONICET Rosario]
Jonathan Protzenko [Microsoft Research]
Eric Tanter [University of Chile, from Mar 2019 until Aug 2019]
Jean-Karim Zinzindohoué [Ministère de l'Intérieur, until Jun 2019]

2. Overall Objectives

2.1. Programming securely with cryptography

In recent years, an increasing amount of sensitive data is being generated, manipulated, and accessed online, from bank accounts to health records. Both national security and individual privacy have come to rely on the security of web-based software applications. But even a single design flaw or implementation bug in an application may be exploited by a malicious criminal to steal, modify, or forge the private records of innocent users. Such *attacks* are becoming increasingly common and now affect millions of users every year.

The risks of deploying insecure software are too great to tolerate anything less than mathematical proof, but applications have become too large for security experts to examine by hand, and automated verification tools do not scale. Today, there is not a single widely-used web application for which we can give a proof of security, even against a small class of attacks. In fact, design and implementation flaws are still found in widely-distributed and thoroughly-vetted security libraries designed and implemented by experts.

Software security is in crisis. A focused research effort is needed if security programming and analysis techniques are to keep up with the rapid development and deployment of security-critical distributed applications based on new cryptographic protocols and secure hardware devices. The goal of our team PROSECCO is to draw upon our expertise in cryptographic protocols and program verification to make decisive contributions in this direction.

Our vision is that, over its lifetime, PROSECCO will contribute to making the use of formal techniques when programming with cryptography as natural as the use of a software debugger. To this end, our long-term goals are to design and implement programming language abstractions, cryptographic models, verification tools, and verified security libraries that developers can use to deploy provably secure distributed applications. Our target applications include cryptographic protocol implementations, hardware-based security APIs, smartphone- and browser-based web applications, and cloud-based web services. In particular, we aim to verify the full application: both the cryptographic core and the high-level application code. We aim to verify implementations, not just models. We aim to account for computational cryptography, not just its symbolic abstraction.

We identify five key focus areas for our research in the short- to medium term.

2.1.1. New programming languages for verified software

Building realistic verified applications requires new programming languages that enable the systematic development of efficient software hand-in-hand with their proofs of correctness. Our current focus is on designing and implementing the programming language F*, in collaboration with Microsoft Research. F* (pronounced F star) is a general-purpose functional programming language with effects aimed at program verification. Its type system includes polymorphism, dependent types, monadic effects, refinement types, and a weakest precondition calculus. Together, these features allow expressing precise and compact specifications for programs, including functional correctness and security properties. The F* type-checker aims to prove that programs meet their specifications using a combination of SMT solving and interactive proofs. Programs written in F* can be translated to efficient OCaml, F#, or C for execution. The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project Everest (a larger collaboration with Microsoft Research). This includes verified implementations of TLS 1.2 and 1.3 and of the underlying cryptographic primitives.

2.1.2. Symbolic verification of cryptographic applications

We aim to develop our own security verification tools for models and implementations of cryptographic protocols and security APIs using symbolic cryptography. Our starting point is the tools we have previously developed: the specialized cryptographic prover ProVerif, the reverse engineering and formal test tool Tookan, and the F* verification system. These tools are already used to verify industrial-strength cryptographic protocol implementations and commercial cryptographic hardware. We plan to extend and combine these approaches to capture more sophisticated attacks on applications consisiting of protocols, software, and hardware, as well as to prove symbolic security properties for such composite systems.

2.1.3. Computational verification of cryptographic applications

We aim to develop our own cryptographic application verification tools that use the computational model of cryptography. The tools include the computational prover CryptoVerif, and the F* verification system. Working together, we plan to extend these tools to analyze, for the first time, cryptographic protocols, security APIs, and their implementations under fully precise cryptographic assumptions. We also plan to pursue links between symbolic and computational verification, such as computational soundness results that enable computational proofs by symbolic techniques.

2.1.4. Efficient formally secure compilers for tagged architectures

We aim to leverage emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilation chains for realistic low-level programming languages (the C language, and Low* a safe subset of C embedded in F* for verification). These compilation chains 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 compilation chains target a tagged architecture, which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules.

2.1.5. Building provably secure web applications

We aim to develop analysis tools and verified libraries to help programmers build provably secure web applications. The tools will include static and dynamic verification tools for client- and server-side JavaScript web applications, their verified deployment within HTML5 websites and browser extensions, as well as type-preserving compilers from high-level applications written in F* to JavaScript. In addition, we plan to model new security APIs in browsers and smartphones and develop the first formal semantics for various HTML5 web standards. We plan to combine these tools and models to analyze the security of multi-party web applications, consisting of clients on browsers and smartphones, and servers in the cloud.

3. Research Program

3.1. Symbolic verification of cryptographic applications

Despite decades of experience, designing and implementing cryptographic applications remains dangerously error-prone, even for experts. This is partly because cryptographic security is an inherently hard problem, and partly because automated verification tools require carefully-crafted inputs and are not widely applicable. To take just the example of TLS, a widely-deployed and well-studied cryptographic protocol designed, implemented, and verified by security experts, the lack of a formal proof about all its details has regularly led to the discovery of major attacks (including several in PROSECCO) on both the protocol and its implementations, after many years of unsuspecting use.

As a result, the automated verification for cryptographic applications is an active area of research, with a wide variety of tools being employed for verifying different kinds of applications.

In previous work, we have developed the following three approaches:

- ProVerif: a symbolic prover for cryptographic protocol models
- Tookan: an attack-finder for PKCS#11 hardware security devices
- F*: a new language that enables the verification of cryptographic applications

3.1.1. Verifying cryptographic protocols with ProVerif

Given a model of a cryptographic protocol, the problem is to verify that an active attacker, possibly with access to some cryptographic keys but unable to guess other secrets, cannot thwart security goals such as authentication and secrecy [53]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [49] and eventually leading to effective verification tools, such as our tool ProVerif.

To use ProVerif, one encodes a protocol model in a formal language, called the applied pi-calculus, and ProVerif abstracts it to a set of generalized Horn clauses. This abstraction is a small approximation: it just ignores the number of repetitions of each action, so ProVerif is still very precise, more precise than, say, tree automata-based techniques. The price to pay for this precision is that ProVerif does not always terminate; however, it terminates in most cases in practice, and it always terminates on the interesting class of *tagged protocols* [44]. ProVerif can handle a wide variety of cryptographic primitives, defined by rewrite rules or by some equations, and prove a wide variety of security properties: secrecy [42], [30], correspondences (including authentication) [43], and observational equivalences [41]. Observational equivalence means that an adversary cannot distinguish two processes (protocols); equivalences can be used to formalize a wide range of properties, but they are particularly difficult to prove. Even if the class of equivalences that ProVerif can prove is limited to equivalences between processes that differ only by the terms they contain, these equivalences are useful in practice and ProVerif has long been the only tool that proves equivalences for an unbounded number of sessions. (Maude-NPA in 2014 and Tamarin in 2015 adopted ProVerif's approach to proving equivalences.)

Using ProVerif, it is now possible to verify large parts of industrial-strength protocols, such as TLS [36], Signal [51], JFK [31], and Web Services Security [40], against powerful adversaries that can run an unlimited number of protocol sessions, for strong security properties expressed as correspondence queries or equivalence assertions. ProVerif is used by many teams at the international level, and has been used in more than 120 research papers (references available at http://proverif.inria.fr/proverif-users.html).

3.1.2. Verifying security APIs using Tookan

Security application programming interfaces (APIs) are interfaces that provide access to functionality while also enforcing a security policy, so that even if a malicious program makes calls to the interface, certain security properties will continue to hold. They are used, for example, by cryptographic devices such as smartcards and Hardware Security Modules (HSMs) to manage keys and provide access to cryptographic functions whilst keeping the keys secure. Like security protocols, their design is security critical and very difficult to get right. Hence formal techniques have been adapted from security protocols to security APIs.

The most widely used standard for cryptographic APIs is RSA PKCS#11, ubiquitous in devices from smartcards to HSMs. A 2003 paper highlighted possible flaws in PKCS#11 [46], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [47]. However at this point it was not clear to what extent these flaws affected real commercial devices, since the standard is underspecified and can be implemented in many different ways. The Tookan tool, developed by Steel in collaboration with Bortolozzo, Centenaro and Focardi, was designed to address this problem. Tookan can reverse engineer the particular configuration of PKCS#11 used by a device under test by sending a carefully designed series of PKCS#11 commands and observing the return codes. These codes are used to instantiate a Dolev-Yao model of the device's API. This model can then be searched using a security protocol model checking tool to find attacks. If an attack is found, Tookan converts the trace from the model checker into the sequence of PKCS#11 queries needed to make the attack and executes the commands directly on the device. Results obtained by Tookan are remarkable: of 18 commercially available PKCS#11 devices tested, 10 were found to be susceptible to at least one attack.

*3.1.3. Verifying cryptographic applications using F**

Verifying the implementation of a protocol has traditionally been considered much harder than verifying its model. This is mainly because implementations have to consider real-world details of the protocol, such as message formats [55], that models typically ignore. So even a protocol has been proved secure in theory, its implementation may be buggy and insecure. However, with recent advances in both program verification and symbolic protocol verification tools, it has become possible to verify fully functional protocol implementations in the symbolic model. One approach is to extract a symbolic protocol model from an implementation and then verify the model, say, using ProVerif. This approach has been quite successful, yielding a verified implementation of TLS in F# [39]. However, the generated models are typically quite large and whole-program symbolic verification does not scale very well.

An alternate approach is to develop a verification method directly for implementation code, using well-known program verification techniques. Our current focus is on designing and implementing the programming language F* [57], [34], [52], in collaboration with Microsoft Research. F* (pronounced F star) is an ML-like functional programming language aimed at program verification. Its type system includes polymorphism, dependent types, monadic effects, refinement types, and a weakest precondition calculus. Together, these features allow expressing precise and compact specifications for programs, including functional correctness and security properties. The F* type-checker aims to prove that programs meet their specifications using a combination of SMT solving and interactive proofs[23]. Programs written in F* can be translated to efficient OCaml, F#, or C for execution [54]. The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project Everest [37] (a larger collaboration with Microsoft Research). This includes a verified implementation of TLS 1.2 and 1.3 [38] and of the underlying cryptographic primitives [58].

3.2. Computational verification of cryptographic applications

Proofs done by cryptographers in the computational model are mostly manual. Our goal is to provide computer support to build or verify these proofs. In order to reach this goal, we have designed the automatic tool CryptoVerif, which generates proofs by sequences of games. We already applied it to important protocols such as TLS [36] and Signal [51] but more work is still needed in order to develop this approach, so that it is easier to apply to more protocols. We also design and implement techniques for proving implementations of protocols secure in the computational model. In particular, CryptoVerif can generate implementations from CryptoVerif specifications that have been proved secure [45]. We plan to continue working on this approach.

A different approach is to directly verify cryptographic applications in the computational model by typing. A recent work [50] shows how to use refinement typechecking in F7 to prove computational security for protocol implementations. In this method, henceforth referred to as computational F7, typechecking is used as the main step to justify a classic game-hopping proof of computational security. The correctness of this method is based on a probabilistic semantics of F# programs and crucially relies on uses of type abstraction and parametricity to establish strong security properties, such as indistinguishability.

In principle, the two approaches, typechecking and game-based proofs, are complementary. Understanding how to combine these approaches remains an open and active topic of research.

An alternative to direct computation proofs is to identify the cryptographic assumptions under which symbolic proofs, which are typically easier to derive automatically, can be mapped to computational proofs. This line of research is sometimes called computational soundness and the extent of its applicability to real-world cryptographic protocols is an active area of investigation.

3.3. F*: A Higher-Order Effectful Language for Program Verification

F* [57], [34] is a verification system for effectful programs developed collaboratively by Inria and Microsoft Research. It puts together the automation of an SMT-backed deductive verification tool with the expressive power of a proof assistant based on dependent types. After verification, F* programs can be extracted to efficient OCaml, F#, or C code [54]. This enables verifying the functional correctness and security of realistic applications. F*'s type system includes dependent types, monadic effects, refinement types, and a weakest precondition calculus. Together, these features allow expressing precise and compact specifications for programs, including functional correctness and security properties. The F* type-checker aims to prove that programs meet their specifications using a combination of SMT solving and interactive proofs. The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project Everest. This includes verified implementations of TLS 1.2 and 1.3 [38] and of the underlying cryptographic primitives [58].

3.4. Efficient Formally Secure Compilers to a Tagged Architecture

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.

We aim to leverage emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilation chains for realistic low-level programming languages (the C language, and Low* a safe subset of C embedded in F* for verification [54]). These compilation chains 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 compilation chains target a tagged architecture [35], which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We hope to experimentally evaluate and carefully optimize the efficiency of our secure compilation chains on realistic workloads and standard benchmark suites. We are also using property-based testing and formal verification to provide high confidence that our compilation chains are indeed secure. Formally, we are constructing machine-checked proofs of a new security criterion we call robustly safe compilation, which is defined as the preservation of safety properties even against an adversarial context [32], [33]. This strong criterion complements compiler correctness and ensures that no machine-code attacker can do more harm to securely compiled components than a component already could with respect to a secure source-level semantics.

3.5. Provably secure web applications

Web applications are fast becoming the dominant programming platform for new software, probably because they offer a quick and easy way for developers to deploy and sell their *app*s to a large number of customers. Third-party web-based apps for Facebook, Apple, and Google, already number in the hundreds of thousands and are likely to grow in number. Many of these applications store and manage private user data, such as health information, credit card data, and GPS locations. To protect this data, applications tend to use an ad hoc combination of cryptographic primitives and protocols. Since designing cryptographic applications is easy to get wrong even for experts, we believe this is an opportune moment to develop security libraries and verification techniques to help web application programmers.

As a typical example, consider commercial password managers, such as LastPass, RoboForm, and 1Password. They are implemented as browser-based web applications that, for a monthly fee, offer to store a user's passwords securely on the web and synchronize them across all of the user's computers and smartphones. The passwords are encrypted using a master password (known only to the user) and stored in the cloud. Hence, no-one except the user should ever be able to read her passwords. When the user visits a web page that has a login form, the password manager asks the user to decrypt her password for this website and automatically fills in the login form. Hence, the user no longer has to remember passwords (except her master password) and all her passwords are available on every computer she uses.

Password managers are available as browser extensions for mainstream browsers such as Firefox, Chrome, and Internet Explorer, and as downloadable apps for Android and Apple phones. So, seen as a distributed application, each password manager application consists of a web service (written in PHP or Java), some number of browser extensions (written in JavaScript), and some smartphone apps (written in Java or Objective C). Each of these components uses a different cryptographic library to encrypt and decrypt password data. How do we verify the correctness of all these components?

We propose three approaches. For client-side web applications and browser extensions written in JavaScript, we propose to build a static and dynamic program analysis framework to verify security invariants. To this end, we have developed two security-oriented type systems for JavaScript, Defensive JavaScript [48] and TS* [56], and used them to guarantee security properties for a number of JavaScript applications. For Android smartphone apps and web services written in Java, we propose to develop annotated JML cryptography libraries that can be used with static analysis tools like ESC/Java to verify the security of application code. For clients and web services written in F# for the .NET platform, we propose to use F* to verify their correctness.

We also propose to translate verified F* web applications to JavaScript via a verified compiler that preserves the semantics of F* programs in JavaScript.

3.6. Design and Verification of next-generation protocols: identity, blockchains, and messaging

Building on our work on verifying and re-designing pre-existing protocols like TLS and Web Security in general, with the resources provided by the NEXTLEAP project, we are working on both designing and verifying new protocols in rapidly emerging areas like identity, blockchains, and secure messaging. These are all areas where existing protocols, such as the heavily used OAuth protocol, are in need of considerable re-design in order to maintain privacy and security properties. Other emerging areas, such as blockchains and secure messaging, can have modifications to existing pre-standard proposals or even a complete 'clean slate' design. As shown by Prosecco's work, newer standards, such as IETF OAuth, W3C Web Crypto, and W3C Web Authentication API, can have vulnerabilities fixed before standardization is complete and heavily deployed. We hope that the tools used by Prosecco can shape the design of new protocols even before they are shipped to standards bodies. We have seen considerable progress in identity with the UnlimitID design and with messaging via the IETF MLS effort, with new work on blockchain technology underway.

4. Application Domains

4.1. Cryptographic Protocol Libraries

Cryptographic protocols such as TLS, SSH, IPSec, and Kerberos are the trusted base on which the security of modern distributed systems is built. Our work enables the analysis and verification of such protocols, both in their design and implementation. Hence, for example, we build and verify models and reference implementations for well-known protocols such as TLS and SSH, as well as analyze their popular implementations such as OpenSSL.

4.2. Hardware-based security APIs

Cryptographic devices such as Hardware Security Modules (HSMs) and smartcards are used to protect long-terms secrets in tamper-proof hardware, so that even attackers who gain physical access to the device cannot obtain its secrets. These devices are used in a variety of scenarios ranging from bank servers to transportation cards (e.g. Navigo). Our work investigates the security of commercial cryptographic hardware and evaluates the APIs they seek to implement.

4.3. Web application security

Web applications use a variety of cryptographic techniques to securely store and exchange sensitive data for their users. For example, a website may serve pages over HTTPS, authenticate users with a single sign-on protocol such as OAuth, encrypt user files on the server-side using XML encryption, and deploy client-side cryptographic mechanisms using a JavaScript cryptographic library. The security of these applications depends on the public key infrastructure (X.509 certificates), web browsers' implementation of HTTPS and the same origin policy (SOP), the semantics of JavaScript, HTML5, and their various associated security standards, as well as the correctness of the specific web application code of interest. We build analysis tools to find bugs in all these artifacts and verification tools that can analyze commercial web applications and evaluate their security against sophisticated web-based attacks.

5. Highlights of the Year

5.1. Highlights of the Year

- We published 12 papers at top-tier conferences and journals such as S&P (1), POPL (2), Euro S&P (2), ICFP (3), CSF (1), ESOP (1)
- Our cryptographic library HACL* was incorporated within the Linux kernel, Microsoft WinQuic, mbedTLS, and Concordium, in addition to the prior deployments in Mozilla Firefox and Tezos Blockchain
- Catalin Hritcu served as Program Chair of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)

5.1.1. Awards

- EU Horizon Impact Award 2019 for Karthikeyan Bhargavan for his research on TLS 1.3.
- Distinguished Paper Award at CSF'19 for "Journey Beyond Full Abstraction"
- Distinguished Paper Award at POPL'19 for "Gradual Parametricity, Revisited"

BEST PAPERS AWARDS:

[17]
C. ABATE, R. BLANCO, D. GARG, C. HRIŢCU, M. PATRIGNANI, J. THIBAULT. *Journey Beyond Full Abstraction: Exploring Robust Property Preservation for Secure Compilation*, in "CSF 2019 - 32nd IEEE Computer Security Foundations Symposium", Hoboken, United States, IEEE, June 2019, pp. 256-271, https://arxiv.org/abs/1807.04603 [*DOI :* 10.1109/CSF.2019.00025], https://hal.archives-ouvertes.fr/hal-02398915

[16]
M. TORO, E. LABRADA, É. TANTER. *Gradual Parametricity, Revisited*, in "Proceedings of the ACM on Programming Languages", 2019, vol. 3, n^{o} POPL, https://arxiv.org/abs/1807.04596 [*DOI :* 10.1145/3290330], https://hal.archives-ouvertes.fr/hal-01960553

6. New Software and Platforms

6.1. Cryptosense Analyzer

SCIENTIFIC DESCRIPTION: Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards, security tokens and Hardware Security Modules that support the most widely-used industry standard interface, RSA PKCS#11. Each device implements PKCS#11 in a slightly different way since the standard is quite open, but finding a subset of the standard that results in a secure device, i.e. one where cryptographic keys cannot be revealed in clear, is actually rather tricky. Cryptosense Analyzer analyses a device by first reverse engineering the exact implementation of PKCS#11 in use, then building a logical model of this implementation for a model checker, calling a model checker to search for attacks, and in the case where an attack is found, executing it directly on the device. It has been used to find at least a dozen previously unknown flaws in commercially available devices.

FUNCTIONAL DESCRIPTION: Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards,

- Participants: Graham Steel and Romain Bardou
- Contact: Graham Steel
- URL: https://cryptosense.com/

6.2. CryptoVerif

Cryptographic protocol verifier in the computational model

KEYWORDS: Security - Verification - Cryptographic protocol

FUNCTIONAL DESCRIPTION: CryptoVerif is an automatic protocol prover sound in the computational model. In this model, messages are bitstrings and the adversary is a polynomial-time probabilistic Turing machine. CryptoVerif can prove secrecy and correspondences, which include in particular authentication. It provides a generic mechanism for specifying the security assumptions on cryptographic primitives, which can handle in particular symmetric encryption, message authentication codes, public-key encryption, signatures, hash functions, and Diffie-Hellman key agreements. It also provides an explicit formula that gives the probability of breaking the protocol as a function of the probability of breaking each primitives, this is the exact security framework.

NEWS OF THE YEAR: We implemented the following features in CryptoVerif:

1) We added to the library of cryptographic primitives several variants of the PRF-ODH (pseudo-random function oracle Diffie-Hellman) assumption, pre-image resistant and second-preimage resistant hash functions, IND-CPA encryption with a nonce, IND-CPA and INT-CTXT encryption with a nonce, encryption schemes that satisfy IND$-CPA instead of IND-CPA.

2) To facilitate modular proofs, we allow querying indistinguishability properties with exactly the same syntax as the one used to specify indistinguishability assumptions on primitives.

3) To simplify declarations of assumptions on primitives, replications (which model any number of copies of processes or oracles) can be omitted at the root of indistinguishability assumptions. CryptoVerif adds them internally, thus inferring the assumption for N independent copies from the assumption for one copy. For instance, it infers the assumption for encryption with N keys from the assumption for encryption with a single key.

4) When we delay random number generations, we allow the user to specify expressions for which it is not necessary to generate the random value, so that the generation of the moved random value can be delayed further. In particular, we used this extension to prove that the OAEP scheme is IND-CCA2 assuming the underlying permutation is partial-domain one-way (a famous cryptographic result).

5) CryptoVerif can now remove parts of the code cannot be executed in case the adversary wins the game, by replacing them with event "AdversaryLoses". That is specially helpful in order to deal with complex cases of key compromise, e.g. for forward secrecy, by proving authentication by ignoring the compromise, showing that authentication is preserved in case the key is compromised (because the adversary never wins against the considered authentication property in case of compromise), and using the authentication to prove secrecy even in case of compromise. For instance, that allows us to show that the PSK-DHE handshake of TLS 1.3 preserves forward secrecy in case of compromise of the PSK.

6) After a cryptographic transformation, CryptoVerif expands terms into processes, which leads to duplicating code until the end of the protocol for each test that is expanded. The cryptographic transformation and the expansion were initially considered as a single transformation. There are now considered as separate transformations, so that other transformations can be performed in between, in particular to cut some branches of the code and reduce the code duplication.

These changes are included in CryptoVerif version 2.02 available at https://cryptoverif.inria.fr.

- Participants: Bruno Blanchet and David Cadé
- Contact: Bruno Blanchet
- Publications: Composition Theorems for CryptoVerif and Application to TLS 1.3 - Composition Theorems for CryptoVerif and Application to TLS 1.3 - A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol - A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol - Proved Implementations of Cryptographic Protocols in the Computational Model - Proved Generation of Implementations from Computationally Secure

Protocol Specifications - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols - Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach

- URL: http://cryptoverif.inria.fr/

6.3. F*

FStar

KEYWORDS: Programming language - Software Verification

FUNCTIONAL DESCRIPTION: F* is a new higher order, effectful programming language (like ML) designed with program verification in mind. Its type system is based on a core that resembles System Fw (hence the name), but is extended with dependent types, refined monadic effects, refinement types, and higher kinds. Together, these features allow expressing precise and compact specifications for programs, including functional correctness properties. The F* type-checker aims to prove that programs meet their specifications using an automated theorem prover (usually Z3) behind the scenes to discharge proof obligations. Programs written in F* can be translated to OCaml, F#, or JavaScript for execution.

- Participants: Antoine Delignat-Lavaud, Catalin Hritcu, Cedric Fournet, Chantal Keller, Karthikeyan Bhargavan and Pierre-Yves Strub
- Contact: Catalin Hritcu
- URL: https://www.fstar-lang.org/

6.4. miTLS

KEYWORDS: Cryptographic protocol - Software Verification

FUNCTIONAL DESCRIPTION: miTLS is a verified reference implementation of the TLS protocol. Our code fully supports its wire formats, ciphersuites, sessions and connections, re-handshakes and resumptions, alerts and errors, and data fragmentation, as prescribed in the RFCs, it interoperates with mainstream web browsers and servers. At the same time, our code is carefully structured to enable its modular, automated verification, from its main API down to computational assumptions on its cryptographic algorithms.

- Participants: Alfredo Pironti, Antoine Delignat-Lavaud, Cedric Fournet, Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Pierre-Yves Strub and Santiago Zanella
- Contact: Karthikeyan Bhargavan
- URL: https://github.com/mitls/mitls-fstar

6.5. ProVerif

KEYWORDS: Security - Verification - Cryptographic protocol

FUNCTIONAL DESCRIPTION: ProVerif is an automatic security protocol verifier in the symbolic model (so called Dolev-Yao model). In this model, cryptographic primitives are considered as black boxes. This protocol verifier is based on an abstract representation of the protocol by Horn clauses. Its main features are:

It can verify various security properties (secrecy, authentication, process equivalences).

It can handle many different cryptographic primitives, specified as rewrite rules or as equations.

It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space.

NEWS OF THE YEAR: Vincent Cheval and Bruno Blanchet worked on several extensions of ProVerif: 1) support for integer counters, with incrementation and inequality tests, 2) lemmas and axioms to give intermediate results to ProVerif, which it exploits to help proving subsequent queries, by deriving additional information in the Horn clauses that it uses to perform the proofs, 3) proofs by induction on the length of the trace, by giving as lemma the property to prove, but obviously for strictly shorter traces. Detailed soundness proofs for these extensions are in progress. These features are not released yet.

- Participants: Bruno Blanchet, Marc Sylvestre and Vincent Cheval
- Contact: Bruno Blanchet
- Publications: Automated reasoning for equivalences in the applied pi calculus with barriers - Automated Reasoning for Equivalences in the Applied Pi Calculus with Barriers - Automated reasoning for equivalences in the applied pi calculus with barriers - Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif - Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols
- URL: http://proverif.inria.fr/

6.6. HACL*

High Assurance Cryptography Library

KEYWORDS: Cryptography - Software Verification

FUNCTIONAL DESCRIPTION: HACL* is a formally verified cryptographic library in F*, developed by the Prosecco team at Inria Paris in collaboration with Microsoft Research, as part of Project Everest.

HACL stands for High-Assurance Cryptographic Library and its design is inspired by discussions at the HACS series of workshops. The goal of this library is to develop verified C reference implementations for popular cryptographic primitives and to verify them for memory safety, functional correctness, and secret independence.

- Contact: Karthikeyan Bhargavan
- URL: https://github.com/mitls/hacl-star

7. New Results

7.1. Verification of security protocols

Participants: Bruno Blanchet, Karthikeyan Bhargavan, Benjamin Lipp.

Our verification of the WireGuard open source Virtual Private Network (VPN) with CryptoVerif appears at EuroS&P 2019 [22], [27].

We continued the development of our protocol verification tools ProVerif and CryptoVerif. The new features of this year are detailed in the section on software.

In the setting of the ANR AnaStaSec project, we worked on the verification of avionic security protocols. More specifically, in 2015, we had verified the protocol of the Secure Dialog Service using ProVerif and CryptoVerif and recommended many changes to the specification. The ICAO started to take into account our remarks, and this year we analyzed a new version of the specification. Our analysis showed that many recommendations still need to be taken into account. Additionally, we also commented on the recent choice of using DTLS over UDP to secure the future ATN/IPS (Aeronautical Telecommunication Network / Internet Protocol Suite) network, which seems very positive. The details of these results are still confidential; they have been provided to ANR.

7.2. Verified Software for Cryptographic Web Applications

Participants: Karthikeyan Bhargavan, Benjamin Beurdouche, Denis Merigoux, Jonathan Protzenko.

WebAssembly in a new language runtime that is now supported by all major web browsers and web application frameworks. We developed a compiler from the Low* subset of the F* programming language to WebAssembly and used this compiler to translate our HACL* verified cryptographic library to WebAssembly, hence obtaining the first verified cryptographic library for the Web. We also used this framework to develop and verify an implementation of the Signal protocol in WebAssembly, and demonstrated how this implementation can be used as a drop-in replacement for the libsignal-protocol library used in mainstream messaging applications like Signal, WhatsApp, and Skype.

Our work was published at the IEEE Security and Privacy conference [24]. Our WebAssembly version of HACL* and our verified Signal implementation were publicly released as open source on GitHub.

7.3. Journey beyond full abstraction

Participants: Carmine Abate, Roberto Blanco, Deepak Garg [MPI-SWS], Catalin Hritcu, Marco Patrignani [Stanford and CISPA], Jérémy Thibault.

Even for safe languages, all guarantees are lost when interacting with low-level code, for instance when using low-level libraries. A compromised or malicious library that gets linked in can easily read and write data and code, jump to arbitrary instructions, or smash the stack, blatantly violating any source-level abstraction and breaking any guarantee obtained by source-level reasoning. Our goal is to build formally secure compartmentalizing compilation chains that defend against such attacks. We started by investigating what it means for a compilation chain to provide secure interoperability between a safe source language and linked target-level code that is adversarial. In this model, a secure compilation chain ensures that even linked adversarial target-level code cannot break the security properties of a compiled program any more than some linked source-level code could. However, the precise class of security properties one chooses to preserve crucially impacts not only the supported security goals and the strength of the attacker model, but also the kind of protections the compilation chain has to introduce and the kind of proof techniques one can use to make sure that the protections are watertight. We are the first to thoroughly explore a large space of secure compilation criteria based on the preservation against adversarial contexts of various classes of trace properties such as safety, of hyperproperties such as noninterference, and of relational hyperproperties such as trace equivalence [17], [10].

7.4. Principles of Program Verification for Arbitrary Monadic Effects

Participants: Kenji Maillard, Danel Ahman [University of Ljubljana], Robert Atkey [University of Strathclyde], Guido Martinez, Catalin Hritcu, Exequiel Rivas, Éric Tanter, Antoine Van Muylder, Cezar Andrici.

We devised a principled semantic framework for verifying programs with arbitrary monadic effects in a generic way with respect to expressive specifications. The starting point are Dijkstra monads, which are monad-like structures that classify effectful computations satisfying a specification drawn from a monad. Dijkstra monads have already proven valuable in practice for verifying effectful code, and in particular, they allow the F* program verifier to compute verification conditions.

We provide the first semantic investigation of the algebraic structure underlying Dijkstra monads [13], [11] and unveil a close relationship between Dijkstra monads and effect observations, i.e., mappings between a computational and a specification monad that respect their monadic structure. Effect observations are flexible enough to provide various interpretations of effects, for instance total vs partial correctness, or angelic vs demonic nondeterminism. Our semantic investigation relies on a general theory of specification monads and effect observations, using an enriched notion of relative monads and relative monad morphisms. We moreover show that a large variety of specification monads can be obtained by applying monad transformers to various base specification monads, including predicate transformers and Hoare-style pre- and postconditions. For defining correct monad transformers, we design a language inspired by the categorical analysis of the relationship between monad transformers and algebras for a monad.

We also adapt our framework to relational verification [14], [11], i.e., proving relational properties between multiple runs of one or more programs, such as noninterference or program equivalence. For this we extend specification monads and effect observations to the relational setting and use them to derive the semantics and core rules of a relational program logic generically for any monadic effect. Finally, we identify and overcome conceptual challenges that prevented previous relational program logics from properly dealing with effects such as exceptions, and are the first to provide a proper semantic foundation and a relational program logic for exceptions.

7.5. Meta-F*: Proof automation with SMT, Tactics, and Metaprograms

Participants: Guido Martinez, Danel Ahman, Victor Dumitrescu, Nick Giannarakis [Princeton University], Chris Hawblitzel [Microsoft Research], Catalin Hritcu, Monal Narasimhamurthy [University of Colorado Boulder], Zoe Paraskevopoulou [Princeton University], Clément Pit-Claudel [MIT], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research].

We introduced Meta-F*[23], a tactics and metaprogramming framework for the F* program verifier. The main novelty of Meta-F* is allowing to use tactics and metaprogramming to discharge assertions not solvable by SMT, or to just simplify them into well-behaved SMT fragments. Plus, Meta-F* can be used to generate verified code automatically.

Meta-F* is implemented as an F* effect, which, given the powerful effect system of F*, heavily increases code reuse and even enables the lightweight verification of metaprograms. Metaprograms can be either interpreted, or compiled to efficient native code that can be dynamically loaded into the F* type-checker and can interoperate with interpreted code. Evaluation on realistic case studies shows that Meta-F* provides substantial gains in proof development, efficiency, and robustness.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Grants with Industry

8.1.1. Evolution, Semantics, and Engineering of the F Verification System*

Grant from Nomadic Labs - Inria

PIs: Catalin Hritcu and Exequiel Rivas

Duration: March 2019 - April 2023

Abstract: While the F* verification system shows great promise in practice, many challenging conceptual problems remain to be solved, many of which can directly inform the further evolution and design of the language. Moreover, many engineering challenges remain in order to build a truly usable verification system. This proposal promises to help address this by focusing on the following 5 main topics: (1) *Generalizing Dijkstra monads*, i.e., a program verification technique for arbitrary monadic effects; (2) *Relational reasoning in F**: devising scalable verification techniques for properties of multiple program executions (e.g., confidentiality, noninterference) or of multiple programs (e.g., program equivalence); (3) *Making F*'s effect system more flexible*, by supporting tractable forms of effect polymorphism and allowing some of the effects of a computation to be hidden if they do not impact the observable behavior; (4) Working out more of the *F* semantics and metatheory*; (5) Solving the *engineering challenges* of building a usable verification system.

9. Partnerships and Cooperations

9.1. National Initiatives

9.1.1. ANR

9.1.1.1. AnaStaSec

Title: Static Analysis for Security Properties (ANR générique 2014.)

Other partners: Inria Paris/EPI Antique, Inria Rennes/EPI Celtique, Airbus Operations SAS, AMOSSYS, CEA-LIST, TrustInSoft

Duration: January 2015 - September 2019.

Coordinator: Jérôme Féret, EPI Antique, Inria Paris (France)

Participant: Bruno Blanchet

Abstract: The project aims at using automated static analysis techniques for verifying security and confidentiality properties of critical avionics software.

9.1.1.2. AJACS

Title: AJACS: Analyses of JavaScript Applications: Certification and Security

Other partners: Inria-Rennes/Celtique, Inria-Saclay/Toccata, Inria-Sophia Antipolis/INDES, Imperial College London

Duration: October 2014 - March 2019.

Coordinator: Alan Schmitt, Inria (France)

Participants: Karthikeyan Bhargavan, Bruno Blanchet, Nadim Kobeissi

Abstract: The goal of the AJACS project is to provide strong security and privacy guarantees for web application scripts. To this end, we propose to define a mechanized semantics of the full JavaScript language, the most widely used language for the Web, to develop and prove correct analyses for JavaScript programs, and to design and certify security and privacy enforcement mechanisms.

9.1.1.3. SafeTLS

Title: SafeTLS: La sécurisation de l'Internet du futur avec TLS 1.

Other partners: Université Rennes 1, IRMAR, Inria Sophia Antipolis, SGDSN/ANSSI

Duration: October 2016 - September 2020

Coordinator: Pierre-Alain Fouque, Université de Rennes 1 (France)

Participants: Karthikeyan Bhargavan

Abstract: Our project, SafeTLS, addresses the security of both TLS 1.3 and of TLS 1.2 as they are (expected to be) used, in three important ways: (1) A better understanding: We will provide a better understanding of how TLS 1.2 and 1.3 are used in real-world applications; (2) Empowering clients: By developing a tool that will show clients the quality of their TLS connection and inform them of potential security and privacy risks; (3) Analyzing implementations: We will analyze the soundness of current TLS 1.2 implementations and use automated verification to provide a backbone of a secure TLS 1.3 implementation.

9.1.1.4. TECAP

Title: TECAP: Protocol Analysis - Combining Existing Tools (ANR générique 2017.)

Other partners: Inria Nancy/EPI PESTO, Inria Sophia Antipolis/EPI MARELLE, IRISA, LIX, LSV - ENS Cachan.

Duration: January 2018 - December 2021

Coordinator: Vincent Cheval, EPI PESTO, Inria Nancy (France)

Participants: Bruno Blanchet, Benjamin Lipp

Abstract: A large variety of automated verification tools have been developed to prove or find attacks on security protocols. These tools differ in their scope, degree of automation, and attacker models. The aim of this project is to get the best of all these tools, meaning, on the one hand, to improve the theory and implementations of each individual tool towards the strengths of the others and, on the other hand, build bridges that allow the cooperations of the methods/tools. We will focus in this project on the tools CryptoVerif, EasyCrypt, Scary, ProVerif, Tamarin, AKiSs and APTE.

9.2. European Initiatives

9.2.1. FP7 & H2020 Projects

9.2.1.1. ERC Consolidator Grant: CIRCUS

Title: CIRCUS: An end-to-end verification architecture for building Certified Implementations of Robust, Cryptographically Secure web applications

Duration: April 2016 - March 2021

Coordinator: Karthikeyan Bhargavan, Inria

The security of modern web applications depends on a variety of critical components including cryptographic libraries, Transport Layer Security (TLS), browser security mechanisms, and single sign-on protocols. Although these components are widely used, their security guarantees remain poorly understood, leading to subtle bugs and frequent attacks. Rather than fixing one attack at a time, we advocate the use of formal security verification to identify and eliminate entire classes of vulnerabilities in one go.

CIRCUS proposes to take on this challenge, by verifying the end-to-end security of web applications running in mainstream software. The key idea is to identify the core security components of web browsers and servers and replace them by rigorously verified components that offer the same functionality but with robust security guarantees.

9.2.1.2. ERC Starting Grant: SECOMP

Title: SECOMP: Efficient Formally Secure Compilers to a Tagged Architecture

Duration: Jan 2017 - December 2021

Coordinator: Catalin Hritcu, Inria

Abstract: The SECOMP project is aimed at leveraging emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilation chains for realistic low-level programming languages (the C language, and Low* a safe subset of C embedded in F* for verification). These compilation chains 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 compilation chains target a tagged architecture, which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We will use property-based testing and formal verification to provide high confidence that our compilers are indeed secure.

9.2.1.3. NEXTLEAP (304)

Title: NEXTLEAP: NEXT generation Legal Encryption And Privacy

Programm: H2020

Duration: January 2016 - December 2018

Coordinator: Harry Halpin, Inria

Other partners: IMDEA, University College London, CNRS, IRI, and Merlinux

The objective of the NEXTLEAP project is to build the fundamental interdisciplinary internet science necessary to create decentralized, secure, and rights-preserving protocols for the next generation of collective awareness platforms. The long-term goal of NEXTLEAP is to have Europe take the “next leap ahead” of the rest of the world by solving the fundamental challenge of determining how both to scientifically build and how to help citizens and institutions adopt open-source decentralized and privacy-preserving digital social platforms in contrast to proprietary centralized cloud-based services and pervasive surveillance that function at the expense of rights and technological sovereignty.

9.3. International Initiatives

9.3.1. Inria International Partners

9.3.1.1. Informal International Partners

We have a range of long- and short-term collaborations with various universities and research labs. We summarize them by project:

- TLS analysis: Microsoft Research (Cambridge), Mozilla, University of Rennes
- F*: Microsoft Research (Redmond, Cambridge, Bangalore), MSR-Inria, CMU, MIT, University of Ljubljana, Nomadic Labs, Zen Protocol, Princeton University
- SECOMP: MPI-SWS, CISPA, Stanford University, CMU, University of Pennsylvania, Portland State University, University of Virginia, University of Iai
- Micro-Policies: University of Pennsylvania, Portland State University, MIT, Draper Labs, Dover Microsystems

9.3.2. Participation in Other International Programs

9.3.2.1. SSITH/HOPE

Title: Advanced New Hardware Optimized for Policy Enforcement, A New HOPE

Program: DARPA SSITH

Duration: December 2017 - February 2021

Coordinator: Charles Stark Draper Laboratory

Other Participants: Inria Paris, University of Pennsylvania, MIT, Portland State University, Dover Microsystems, DornerWorks

Participants from Inria Prosecco: Catalin Hritcu, Roberto Blanco, Jérémy Thibault

Abstract: A New HOPE builds on results from the Inherently Secure Processor (ISP) project that has been internally funded at Draper. Recent architectural improvements decouple the tagged architecture from the processor pipeline to improve performance and flexibility for new processors. HOPE securely maintains metadata for each word in application memory and checks every instruction against a set of installed security policies. The HOPE security architecture exposes tunable parameters that support Performance, Power, Area, Software compatibility and Security (PPASS) search space exploration. Flexible software-defined security policies cover all 7 SSITH CWE vulnerability classes, and policies can be tuned to meet PPASS requirements; for example, one can trade granularity of security checks against performance using different policy configurations. HOPE will design and formalize a new high-level domain-specific language (DSL) for defining security policies, based on previous research and on extensive experience with previous policy languages. HOPE will formally verify that installed security policies satisfy system-wide security requirements. A secure boot process enables policies to be securely updated on deployed HOPE systems. Security policies can adapt based on previously detected attacks. Over the multi-year, multi-million dollar Draper ISP project, the tagged security architecture approach has evolved from early prototypes based on results from the DARPA CRASH program towards easier integration with external designs, and is better able to scale from micro to server class implementations. A New HOPE team is led by Draper and includes faculty from University of Pennsylvania (Penn), Portland State University (PSU), Inria, and MIT, as well as industry collaborators from DornerWorks and Dover Microsystems. In addition to Draper's in-house expertise in hardware design, cyber-security (defensive and offensive, hardware and software) and formal methods, the HOPE team includes experts from all domains relevant to SSITH, including (a) computer architecture: DeHon (Penn), Shrobe (MIT); (b) formal methods including programming languages and security: Pierce (Penn), Tolmach (PSU), Hritcu (Inria); and (c) operating system integration (DornerWorks). Dover Microsystems is a spin-out from Draper that will commercialize concepts from the Draper ISP project.

9.3.2.2. Everest Expedition

Program: Microsoft Expedition and MSR-Inria Collaborative Research Project

Expedition Participants: Microsoft Research (Cambridge, Redmond, Bangalore), Inria, MSR-Inria, CMU, University of Edinburgh

Duration of current MSR-Inria Project: October 2017 – October 2020

Participants from Inria Prosecco: Karthikeyan Bhargavan, Catalin Hritcu, Danel Ahman, Benjamin Beurdouche, Victor Dumitrescu, Nadim Kobeissi, Théo Laurent, Guido Martínez, Denis Merigoux, Marina Polubelova, Jean-Karim Zinzindohoué

Participants from other Inria teams: David Pichardie (Celtique), Jean-Pierre Talpin (TEA)

Abstract: The HTTPS ecosystem (HTTPS and TLS protocols, X.509 public key infrastructure, crypto algorithms) is the foundation on which Internet security is built. Unfortunately, this ecosystem is brittle, with headline-grabbing attacks such as FREAK and LogJam and emergency patches many times a year.

Project Everest addresses this problem by constructing a high-performance, standards-compliant, formally verified implementation of components in HTTPS ecosystem, including TLS, the main protocol at the heart of HTTPS, as well as the main underlying cryptographic algorithms such as AES, SHA2 or X25519.

At the TLS level, for instance, we are developing new implementations of existing and forthcoming protocol standards and formally proving, by reduction to cryptographic assumptions on their core algorithms, that our implementations provide a secure-channel abstraction between the communicating endpoints. Implementations of the core algorithms themselves are also verified, producing performant portable C code or highly optimized assembly language.

We aim for our verified components to be drop-in replacements suitable for use in mainstream web browsers, servers, and other popular tools and are actively working with the community at large to improve the ecosystem.

https://project-everest.github.io

9.4. International Research Visitors

9.4.1. Visits of International Scientists

- Éric Tanter (University of Chile) joined Inria as a Visiting Professor from Jul 2018 to March 2019 and from August to December 2019; he gave various seminars at Inria including one entitled “Gradual Parametricity, Revisited”;
- Li-yao Xia (University of Pennsylvania) visited Prosecco on 7 January and gave a talk entitled “From C to Interaction Trees”;
- Matías Toro (University of Chile) visited Prosecco on 9 January and gave a talk entitled “Type-Driven Gradual Security with References”;
- Deepak Garg (MPI-SWS) visited Prosecco on 29 January and 20 November;
- Gilles Barthe (MPI-SP) visited Prosecco on various occasions: 29 January, 3–6 June, 9–13 Sept, and 7–9 October 2019;
- Jeremy Siek (Indiana University) visited Prosecco on 21 February and gave a seminar entitled “Toward Efficient Gradual Typing”;
- Andrew Tolmach (Portland State University) visted Prosecco on 8–12 April and gave a seminar on “Enforcing C-level security policies using machine-level tags”;
- Guido Martinez (CIFASIS-CONICET Rosario) visited Prosecco on various occasions: April 15–19, ICFP, 30 September to 12 October

- Nikos Vasilakis (University of Pennsylvania) visited Prosecco on 15–19 July and gave a seminar on "Retrofitting Security, Module by Module";
- Clement Pit-Claudel (MPI) visited Prosecco on 14 August;
- Kevin Liao (MPI-SP) visited Prosecco on various occasions and gave a seminar on "ILC: A Calculus for Composable, Computational Cryptography";
- Tahina Ramananandro (Microsoft Research) visited Prosecco on 30 September to 15 October and gave a seminar on "EverParse";
- Nik Swamy (Microsoft Research) and Aymeric Fromherz (CMU) visited Prosecco from 7–11 October and gave a seminar on "Verifying a mixture of C and assembly code with Low* and Vale";
- Jonathan Protzenko (Microsoft Research) visited Prosecco on 30 September to 15 October and gave a seminar on "The EverCrypt verified cryptographic provider";
- Jakob von Raumer (University of Nottingham) visited Prosecco on 23 October and gave a seminar on "Indexed Inductive Types";
- Bas Spitters (COBRA, Aarhus University) visited Prosecco on 25–29 November and gave a seminar on "ConCert: A Smart Contract Certification Framework in Coq";
- Adrien Koutsos (MPI-SP) visited Prosecco on 5 November and gave a talk on "5G-AKA authentication protocol privacy";
- Akram El-Korashy (MPI-SWS) visited Prosecco on 20 November;
- Shin-ya Katsumata (NII, Tokyo, Japan) visited Prosecco on 25–28 November;
- Ian Miers (Johns Hopkins University) visited Prosecco on 29 November and gave a seminar on "Zcash, Blockchains, and the possibilities for formal verification with zero-knowledge";

9.4.1.1. Internships

- Antoine Van Muylder (Paris 7): from April to September 2019 – advised by Catalin Hritcu, Exequiel Rivas, and Kenji Maillard
- Guillaume Gette: from April to September 2019 – advised by Karthikeyan Bhargavan
- Mikhail Volkhov: from April to August 2019 – advised by Karthikeyan Bhargavan and Prasad Naldurg

9.4.2. Visits to International Teams

- Catalin Hritcu visited EPFL Lausanne on 25–27 September;
- Catalin Hritcu, Carmine Abate, Roberto Blanco, and Jeremy Thibault visited MPI-SWS in Saarbrücken on 18–22 October and 1–3 December;
- Catalin Hritcu visited Chalmers University in Gothenburg on 4–6 December;

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Organisation

10.1.1.1. General Chair, Scientific Chair

- Catalin Hritcu is the Steering Committee Chair of the Workshop on Principles of Secure Compilation (PriSC)
- Catalin Hritcu is the main organizer a Dagstuhl Seminar on Secure Compilation (20201)
- Karthikeyan Bhargavan co-chaired the Workshop on Secure Messaging at EUROCRYPT 2019

10.1.1.2. Member of the Organizing Committees

- Catalin Hritcu is a Steering Committee Member of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)

10.1.2. Scientific Events: Selection

10.1.2.1. Chair of Conference Program Committees

- Catalin Hritcu served as Program Chair of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)

10.1.2.2. Member of the Conference Program Committees

- Karthikeyan Bhargavan was PC member of CCS 2019
- Bruno Blanchet was PC member of CSF 2019
- Catalin Hritcu was PC member of RV 2019, SecDev 2019, CSF 2020

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

Associate Editor

– of the *International Journal of Applied Cryptography (IJACT)* – Inderscience Publishers: Bruno Blanchet

10.1.4. Invited Talks

- Karthikeyan Bhargavan gave invited talks at ERCIM Rome, FSTTCS Mumbai, ICISS Hyderabad,
- Catalin Hritcu gave invited talks at EPFL Lausanne, Ruhr University Bochum, MPI-SWS, and Chalmers University;

10.1.5. Leadership within the Scientific Community

- Catalin Hritcu served as the Artifact Evaluation Co-Chair for POPL 2018 and POPL 2019

10.1.6. Scientific Expertise

- Bruno Blanchet is a member of the specialized temporary scientific committee of ANSM (*Agence nationale de sécurité du médicament et des produits de santé*), on the cybersecurity of software medical devices.
- Bruno Blanchet was a scientific consultant for Nomadic Labs, regarding the development of the blockchain Tezos.
- Karthikeyan Bhargavan was scientific consultant for Nomadic Labs, regarding verified cryptographic software.

10.1.7. Research Administration

- Bruno Blanchet was a member of the hiring scientific jury for Inria researchers (*chargé de recherche*) of the Inria Paris center.
- Bruno Blanchet was a representative of Inria Paris at the DIM RFSI (*Domaine d'Intérêt Majeur, Réseau Francilien en Sciences Informatiques*).

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

- Master: Bruno Blanchet, Cryptographic protocols: formal and computational proofs, 18h equivalent TD, master M2 MPRI, université Paris VII
- Master: Karthikeyan Bhargavan, Cryptographic protocols: formal and computational proofs, 18h equivalent TD, master M2 MPRI, université Paris VII

- PhD: Karthikeyan Bhargavan, Verified Crypto for Verified Protocols, Sibenik Summer School on real-world crypto and privacy, 17-21 June, 2019
- PhD: Catalin Hritcu, Program Verification with F* course at Summer School on Verification Technology, Systems, and Applications, VSTA 2019, 1-5 July 2019 at University of Luxembourg
- PhD: Catalin Hritcu, Writing and Verifying Functional Programs in Coq course at Summer School on Cryptography, Blockchain, and Program Verification, Mathinfoly 2019, 24-31 August 2019 at INSA, Lyon

10.2.2. Supervision

- PhD in progress: Benjamin Lipp, On Mechanised Cryptographic Proofs of Protocols and their Link with Verified Implementations, ENS Paris, since October 2018, supervised by Bruno Blanchet and Karthikeyan Bhargavan.
- PhD in progress: Benjamin Beurdouche, Formal Verification for Real-World Cryptographic Protocols, PSL, since September 2016, supervised by Karthikeyan Bhargavan.
- PhD in progress: Natalia Kulatova, Formal Analysis of Security Devices, PSL, since September 2017, supervised by Karthikeyan Bhargavan and Graham Steel.
- PhD in progress: Marina Polubelova, Formal Verification of a Cryptographic Library, PSL, since September 2017, supervised by Karthikeyan Bhargavan.
- PhD in progress: Denis Merigoux, Verification framework for performance-oriented memory-safe programming languages, since September 2018, supervised by Karthikeyan Bhargavan and Jonathan Protzenko.
- PhD: Kenji Maillard, on Principles of Program Verification for Arbitrary Monadic Effects, started January 2017, supervised by Catalin Hritcu
- PhD in progress: Carmine Abate, The Formal Foundations of Secure Compilation, since June 2018, advised by Catalin Hritcu
- PhD in progress: Jérémy Thibault, Secure Compartmentalizing Compilation to a Tagged Architecture, from August 2018, advised by Catalin Hritcu
- PhD in progress: Guido Martínez (CIFASIS-CONICET Rosario), Metatheory for Semi-Automatic Verification of Effectful Programs, from April 2017, advised by Mauro Jaskelioff (CIFASIS-CONICET Rosario) and Catalin Hritcu

10.2.3. Juries

- Bruno Blanchet was member of the PhD jury of Adrien Koutsos (ENS Paris-Saclay).
- Karthikeyan Bhargavan was member of the PhD jury of Joseph Lallemand (Univ. Lorraine) and Guido Martinez (Univ Stuttgart).
- Catalin Hritcu was a discussion leader for the Licentiate defense of Maximilian Algehed (Chalmers University);

10.3. Popularization

10.3.1. Articles and contents

- Catalin Hritcu contributed an article on Secure Compilation to the SIGPLAN PL Perspectives blog
- Karthikeyan Bhargavan published a paper in Communications of the ACM

11. Bibliography

Major publications by the team in recent years

[1] M. ABADI, B. BLANCHET, C. FOURNET. *The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication*, in "Journal of the ACM (JACM)", October 2017, vol. 65, n^{o} 1, pp. 1 - 103 [*DOI :* 10.1145/3127586], https://hal.inria.fr/hal-01636616

[2] C. ABATE, A. AZEVEDO DE AMORIM, R. BLANCO, A. N. EVANS, G. FACHINI, C. HRIŢCU, T. LAURENT, B. C. PIERCE, M. STRONATI, A. TOLMACH. *When Good Components Go Bad: Formally Secure Compilation Despite Dynamic Compromise*, in "25th ACM Conference on Computer and Communications Security (CCS)", Toronto, Canada, ACM, October 2018, pp. 1351–1368, https://arxiv.org/abs/1802.00588 [*DOI :* 10.1145/3243734.3243745], https://hal.archives-ouvertes.fr/hal-01949202

[3] K. BHARGAVAN, B. BLANCHET, N. KOBEISSI. *Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate*, in "38th IEEE Symposium on Security and Privacy", San Jose, United States, May 2017, pp. 483 - 502 [*DOI :* 10.1109/SP.2017.26], https://hal.inria.fr/hal-01575920

[4] K. BHARGAVAN, A. DELIGNAT-LAVAUD, C. FOURNET, A. PIRONTI, P.-Y. STRUB. *Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS*, in "IEEE Symposium on Security and Privacy (Oakland)", 2014, pp. 98–113, https://hal.inria.fr/hal-01102259

[5] B. BLANCHET. *Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif*, in "Foundations and Trends in Privacy and Security", October 2016, vol. 1, n^{o} 1–2, pp. 1–135, https://hal.inria.fr/hal-01423760

[6] M. ISAAKIDIS, H. HALPIN, G. DANEZIS. *UnlimitID: Privacy-Preserving Federated Identity Management Using Algebraic MACs*, in "Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society", New York, NY, USA, WPES '16, ACM, 2016, pp. 139–142 [*DOI :* 10.1145/2994620.2994637], https://hal.inria.fr/hal-01426847

[7] N. KOBEISSI, K. BHARGAVAN, B. BLANCHET. *Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach*, in "2nd IEEE European Symposium on Security and Privacy", Paris, France, April 2017, pp. 435 - 450 [*DOI :* 10.1109/EUROSP.2017.38], https://hal.inria.fr/hal-01575923

[8] 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, S. ZANELLA-BÉGUELIN. *Dependent Types and Multi-Monadic Effects in F**, in "43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", ACM, January 2016, pp. 256-270, https://hal.inria.fr/hal-01265793

[9] J. K. ZINZINDOHOUÉ, K. BHARGAVAN, J. PROTZENKO, B. BEURDOUCHE. *HACL*: A Verified Modern Cryptographic Library*, in "Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017", 2017, pp. 1789–1806, https://hal.inria.fr/hal-01588421

Publications of the year

Doctoral Dissertations and Habilitation Theses

[10] C. HRIŢCU. *The Quest for Formally Secure Compartmentalizing Compilation*, ENS Paris ; PSL Research University, January 2019, Habilitation à diriger des recherches, https://tel.archives-ouvertes.fr/tel-01995823

[11] K. MAILLARD. *Principles of Program Verification for Arbitrary Monadic Effects*, ENS Paris - Ecole Normale Supérieure de Paris, November 2019, https://hal.archives-ouvertes.fr/tel-02416788

Articles in International Peer-Reviewed Journals

[12] J. EREMONDI, É. TANTER, R. GARCIA. *Approximate normalization for gradual dependent types*, in "Proceedings of the ACM on Programming Languages", July 2019, vol. 3, n° ICFP, pp. 1-30 [*DOI :* 10.1145/3341692], https://hal.archives-ouvertes.fr/hal-02399594

[13] K. MAILLARD, D. AHMAN, R. ATKEY, G. MARTÍNEZ, C. HRIŢCU, E. RIVAS, É. TANTER. *Dijkstra monads for all*, in "Proceedings of the ACM on Programming Languages", July 2019, vol. 3, n° ICFP, pp. 1-29, https://arxiv.org/abs/1903.01237 [*DOI :* 10.1145/3341708], https://hal.archives-ouvertes.fr/hal-02398919

[14] K. MAILLARD, C. HRIŢCU, E. RIVAS, A. VAN MUYLDER. *The Next 700 Relational Program Logics*, in "Proceedings of the ACM on Programming Languages", 2019, vol. 4, n° POPL, https://arxiv.org/abs/1907.05244 , forthcoming, https://hal.archives-ouvertes.fr/hal-02398927

[15] P.-M. PÉDROT, N. TABAREAU, H. J. FEHRMANN, É. TANTER. *A Reasonably Exceptional Type Theory*, in "Proceedings of the ACM on Programming Languages", August 2019, vol. 3, pp. 1-29 [*DOI :* 10.1145/3341712], https://hal.inria.fr/hal-02189128

[16] *Best Paper*
M. TORO, E. LABRADA, É. TANTER. *Gradual Parametricity, Revisited*, in "Proceedings of the ACM on Programming Languages", 2019, vol. 3, n° POPL, https://arxiv.org/abs/1807.04596 [*DOI :* 10.1145/3290330], https://hal.archives-ouvertes.fr/hal-01960553.

International Conferences with Proceedings

[17] *Best Paper*
C. ABATE, R. BLANCO, D. GARG, C. HRIŢCU, M. PATRIGNANI, J. THIBAULT. *Journey Beyond Full Abstraction: Exploring Robust Property Preservation for Secure Compilation*, in "CSF 2019 - 32nd IEEE Computer Security Foundations Symposium", Hoboken, United States, IEEE, June 2019, pp. 256-271, https://arxiv.org/abs/1807.04603 [*DOI :* 10.1109/CSF.2019.00025], https://hal.archives-ouvertes.fr/hal-02398915.

[18] R. BLANCO, D. MILLER, A. MOMIGLIANO. *Property-Based Testing via Proof Reconstruction*, in "PPDP 2019 - 21st International Symposium on Principles and Practice of Programming Languages", Porto, Portugal, ACM Press, October 2019, pp. 1-13 [*DOI :* 10.1145/3354166.3354170], https://hal.inria.fr/hal-02368931

[19] R. CRUZ, É. TANTER. *Polymorphic Relaxed Noninterference*, in "SecDev 2019 : IEEE Secure Development Conference", McLean, VA, United States, IEEE, 2019, pp. 101-113 [*DOI :* 10.1109/SECDEV.2019.00021], https://hal.archives-ouvertes.fr/hal-02399576

[20] T. DÍAZ, F. OLMEDO, É. TANTER. *A Mechanized Formalization of GraphQL*, in "CPP 2020 - 9th ACM SIGPLAN International Conference on Certified Programs and Proofs", New Orleans, United States, January 2020 [*DOI :* 10.1145/3372885.3373822], https://hal.archives-ouvertes.fr/hal-02422532

[21] N. KOBEISSI, G. NICOLAS, K. BHARGAVAN. *Noise Explorer: Fully Automated Modeling and Verification for Arbitrary Noise Protocols*, in "EuroS&P 2019 - 4th IEEE European Symposium on Security and Privacy", Stockholm, Sweden, June 2019, https://hal.inria.fr/hal-01948964

[22] B. LIPP, B. BLANCHET, K. BHARGAVAN. *A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol*, in "4th IEEE European Symposium on Security and Privacy", Stockholm, Sweden, IEEE Computer Society, June 2019, pp. 231-246, https://hal.inria.fr/hal-02396640

[23] 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, N. SWAMY. *Meta-F*: Proof automation with SMT, Tactics, and Metaprograms*, in "ESOP'19 - European Symposium on Programming", Prague, Czech Republic, April 2019, https://arxiv.org/abs/1803.06547 , https://hal.archives-ouvertes.fr/hal-01995376

[24] J. PROTZENKO, B. BEURDOUCHE, D. MERIGOUX, K. BHARGAVAN. *Formally Verified Cryptographic Web Applications in WebAssembly*, in "SP 2019 - 40th IEEE Symposium on Security and Privacy", San Francisco, United States, IEEE, May 2019, pp. 1256-1274 [*DOI :* 10.1109/SP.2019.00064], https://hal.inria.fr/hal-02294935

National Conferences with Proceedings

[25] D. MERIGOUX, R. MONAT, C. GAIE. *Étude formelle de l'implémentation du code des impôts*, in "31ème Journées Francophones des Langages Applicatifs", Gruissan, France, January 2020, https://hal.inria.fr/hal-02320347

Research Reports

[26] K. BHARGAVAN, B. BEURDOUCHE, P. NALDURG. *Formal Models and Verified Protocols for Group Messaging: Attacks and Proofs for IETF MLS*, Inria Paris, December 2019, https://hal.inria.fr/hal-02425229

[27] B. LIPP, B. BLANCHET, K. BHARGAVAN. *A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol*, Inria Paris, April 2019, n° RR-9269, 49 p. , https://hal.inria.fr/hal-02100345

Other Publications

[28] B. BEURDOUCHE. *MLS Architecture: analysis of the security, privacy and functional requirements*, January 2020, working paper or preprint, https://hal.inria.fr/hal-02439526

[29] E. RIVAS, M. JASKELIOFF. *Monads with merging*, June 2019, working paper or preprint, https://hal.inria.fr/hal-02150199

References in notes

[30] M. ABADI, B. BLANCHET. *Analyzing Security Protocols with Secrecy Types and Logic Programs*, in "Journal of the ACM", January 2005, vol. 52, n° 1, pp. 102–146, http://prosecco.gforge.inria.fr/personal/bblanche/publications/AbadiBlanchetJACM7037.pdf

[31] M. ABADI, B. BLANCHET, C. FOURNET. *Just Fast Keying in the Pi Calculus*, in "ACM Transactions on Information and System Security (TISSEC)", July 2007, vol. 10, n° 3, pp. 1–59, http://prosecco.gforge.inria.fr/personal/bblanche/publications/AbadiBlanchetFournetTISSEC07.pdf

[32] C. ABATE, R. BLANCO, D. GARG, C. HRIŢCU, M. PATRIGNANI, J. THIBAULT. *Journey Beyond Full Abstraction: Exploring Robust Property Preservation for Secure Compilation*, in "32nd IEEE Computer

Security Foundations Symposium (CSF)", IEEE, June 2019, pp. 256-271 [*DOI :* 10.1109/CSF.2019.00025], https://arxiv.org/abs/1807.04603

[33] C. ABATE, A. AZEVEDO DE AMORIM, R. BLANCO, A. N. EVANS, G. FACHINI, C. HRIŢCU, T. LAURENT, B. C. PIERCE, M. STRONATI, A. TOLMACH. *When Good Components Go Bad: Formally Secure Compilation Despite Dynamic Compromise*, in "25th ACM Conference on Computer and Communications Security (CCS)", ACM, October 2018, pp. 1351–1368, https://arxiv.org/abs/1802.00588

[34] D. AHMAN, C. HRIŢCU, K. MAILLARD, G. MARTÍNEZ, G. PLOTKIN, J. PROTZENKO, A. RASTOGI, N. SWAMY. *Dijkstra Monads for Free*, in "44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)", ACM, January 2017, pp. 515-529 [*DOI :* 10.1145/3009837.3009878], https://www.fstar-lang.org/papers/dm4free/

[35] A. AZEVEDO DE AMORIM, M. DÉNÈS, N. GIANNARAKIS, C. HRITCU, B. C. PIERCE, A. SPECTOR-ZABUSKY, A. TOLMACH. *Micro-Policies: Formally Verified, Tag-Based Security Monitors*, in "36th IEEE Symposium on Security and Privacy (Oakland S&P)", IEEE Computer Society, May 2015, pp. 813–830 [*DOI :* 10.1109/SP.2015.55], http://prosecco.gforge.inria.fr/personal/hritcu/publications/micro-policies.pdf

[36] K. BHARGAVAN, B. BLANCHET, N. KOBEISSI. *Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate*, in "38th IEEE Symposium on Security and Privacy", San Jose, United States, May 2017, pp. 483 - 502 [*DOI :* 10.1109/SP.2017.26], https://hal.inria.fr/hal-01575920

[37] K. BHARGAVAN, B. BOND, A. DELIGNAT-LAVAUD, C. FOURNET, C. HAWBLITZEL, C. HRIŢCU, S. ISHTIAQ, M. KOHLWEISS, R. LEINO, J. LORCH, K. MAILLARD, J. PAN, B. PARNO, J. PROTZENKO, T. RAMANANANDRO, A. RANE, A. RASTOGI, N. SWAMY, L. THOMPSON, P. WANG, S. ZANELLA-BÉGUELIN, J. K. ZINZINDOHOUÉ. *Everest: Towards a Verified, Drop-in Replacement of HTTPS*, in "2nd Summit on Advances in Programming Languages (SNAPL)", May 2017, http://drops.dagstuhl.de/opus/volltexte/2017/7119/pdf/LIPIcs-SNAPL-2017-1.pdf

[38] K. BHARGAVAN, A. DELIGNAT-LAVAUD, C. FOURNET, M. KOHLWEISS, J. PAN, J. PROTZENKO, A. RASTOGI, N. SWAMY, S. ZANELLA-BÉGUELIN, J. K. ZINZINDOHOUÉ. *Implementing and Proving the TLS 1.3 Record Layer*, in "IEEE Symposium on Security and Privacy (Oakland)", 2017

[39] K. BHARGAVAN, C. FOURNET, R. CORIN, E. ZALINESCU. *Verified Cryptographic Implementations for TLS*, in "ACM Transactions Inf. Syst. Secur.", March 2012, vol. 15, n^{o} 1, pp. 3:1–3:32, http://doi.acm.org/10.1145/2133375.2133378

[40] K. BHARGAVAN, C. FOURNET, A. D. GORDON, N. SWAMY. *Verified implementations of the information card federated identity-management protocol*, in "ACM Symposium on Information, Computer and Communications Security (ASIACCS)", 2008, pp. 123-135

[41] B. BLANCHET, M. ABADI, C. FOURNET. *Automated Verification of Selected Equivalences for Security Protocols*, in "Journal of Logic and Algebraic Programming", February–March 2008, vol. 75, n^{o} 1, pp. 3–51, http://prosecco.gforge.inria.fr/personal/bblanche/publications/BlanchetAbadiFournetJLAP07.pdf

[42] B. BLANCHET. *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules*, in "14th IEEE Computer Security Foundations Workshop (CSFW'01)", 2001, pp. 82–96

[43] B. BLANCHET. *Automatic Verification of Correspondences for Security Protocols*, in "Journal of Computer Security", July 2009, vol. 17, n° 4, pp. 363–434, http://prosecco.gforge.inria.fr/personal/bblanche/publications/BlanchetJCS08.pdf

[44] B. BLANCHET, A. PODELSKI. *Verification of Cryptographic Protocols: Tagging Enforces Termination*, in "Theoretical Computer Science", March 2005, vol. 333, n° 1-2, pp. 67–90, Special issue FoSSaCS'03., http://prosecco.gforge.inria.fr/personal/bblanche/publications/BlanchetPodelskiTCS04.html

[45] D. CADÉ, B. BLANCHET. *Proved Generation of Implementations from Computationally Secure Protocol Specifications*, in "Journal of Computer Security", 2015, vol. 23, n° 3, pp. 331–402

[46] J. CLULOW. *On the Security of PKCS#11*, in "CHES", 2003, pp. 411-425

[47] S. DELAUNE, S. KREMER, G. STEEL. *Formal Analysis of PKCS#11 and Proprietary Extensions*, in "Journal of Computer Security", November 2010, vol. 18, n° 6, pp. 1211-1245 [*DOI :* 10.3233/JCS-2009-0394], http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKS-jcs09.pdf

[48] A. DELIGNAT-LAVAUD, K. BHARGAVAN, S. MAFFEIS. *Language-Based Defenses Against Untrusted Browser Origins*, in "Proceedings of the 22th USENIX Security Symposium", 2013, http://prosecco.inria.fr/personal/karthik/pubs/language-based-defenses-against-untrusted-origins-sec13.pdf

[49] D. DOLEV, A. YAO. *On the security of public key protocols*, in "IEEE Transactions on Information Theory", 1983, vol. IT–29, n° 2, pp. 198–208

[50] C. FOURNET, M. KOHLWEISS, P.-Y. STRUB. *Modular Code-Based Cryptographic Verification*, in "ACM Conference on Computer and Communications Security", 2011

[51] N. KOBEISSI, K. BHARGAVAN, B. BLANCHET. *Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach*, in "2nd IEEE European Symposium on Security and Privacy", Paris, France, April 2017, pp. 435 - 450 [*DOI :* 10.1109/EUROSP.2017.38], https://hal.inria.fr/hal-01575923

[52] K. MAILLARD, D. AHMAN, R. ATKEY, G. MARTÍNEZ, C. HRIŢCU, E. RIVAS, É. TANTER. *Dijkstra Monads for All*, in "PACMPL", 2019, vol. 3, n° ICFP, pp. 104:1–104:29 [*DOI :* 10.1145/3341708], https://arxiv.org/abs/1903.01237

[53] R. NEEDHAM, M. SCHROEDER. *Using encryption for authentication in large networks of computers*, in "Communications of the ACM", 1978, vol. 21, n° 12, pp. 993–999

[54] J. PROTZENKO, J. K. ZINZINDOHOUÉ, A. RASTOGI, T. RAMANANANDRO, P. WANG, S. ZANELLA-BÉGUELIN, A. DELIGNAT-LAVAUD, C. HRIŢCU, K. BHARGAVAN, C. FOURNET, N. SWAMY. *Verified Low-Level Programming Embedded in F**, in "PACMPL", September 2017, vol. 1, n° ICFP, pp. 17:1–17:29 [*DOI :* 10.1145/3110261], http://arxiv.org/abs/1703.00053

[55] T. RAMANANANDRO, A. DELIGNAT-LAVAUD, C. FOURNET, N. SWAMY, T. CHAJED, N. KOBEISSI, J. PROTZENKO. *EverParse: Verified Secure Zero-Copy Parsers for Authenticated Message Formats*, in "28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019", N. HENINGER, P. TRAYNOR (editors), USENIX Association, 2019, pp. 1465–1482, https://www.usenix.org/conference/usenixsecurity19/presentation/delignat-lavaud

[56] N. SWAMY, C. FOURNET, A. RASTOGI, K. BHARGAVAN, J. CHEN, P.-Y. STRUB, G. M. BIERMAN. *Gradual typing embedded securely in JavaScript*, in "41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", 2014, pp. 425-438, http://prosecco.inria.fr/personal/karthik/pubs/tsstar-popl14.pdf

[57] 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, S. ZANELLA-BÉGUELIN. *Dependent Types and Multi-Monadic Effects in F**, in "43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", ACM, January 2016, pp. 256-270, https://www.fstar-lang.org/papers/mumon/

[58] J. K. ZINZINDOHOUÉ, K. BHARGAVAN, J. PROTZENKO, B. BEURDOUCHE. *HACL*: A Verified Modern Cryptographic Library*, in "Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017", 2017, pp. 1789–1806, http://doi.acm.org/10.1145/3133956.3134043