Activity Report 2017

Project-Team PROSECCO

Programming securely with cryptography

RESEARCH CENTER
Paris

THEME
Security and Confidentiality

Table of contents

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
A2.1.3. - Functional programming
A2.1.7. - Distributed programming
A2.1.11. - Proof languages
A2.2. - Compilation
A2.2.1. - Static analysis
A2.2.3. - Run-time systems
A2.4. - 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
B9.8. - Privacy

1. Personnel

Research Scientists

Karthikeyan Bhargavan [Team leader, Inria, Senior Researcher, HDR]
Amal Ahmed [Inria, Advanced Research Position, from Sep 2017]
David Baelde [Ecole Normale Supérieure Cachan, Researcher, until Aug 2017]
Bruno Blanchet [Inria, Senior Researcher, HDR]
Harry Halpin [Inria, Starting Research Position]
Catalin Hritcu [Inria, Researcher]

Post-Doctoral Fellows

Danel Ahman [Inria, from Oct 2017]
Marco Stronati [Inria]

PhD Students

Benjamin Beurdouche [Inria]
Nadim Kobeissi [Inria]
Natalia Kulatova [Inria, from Nov 2017]
Kenji Maillard [Ecole Normale Supérieure Paris]
Marina Polubelova [Inria, from Sep 2017]
Jean-Karim Zinzindohoué [Ministère de l'Ecologie, de l'Energie, du Développement durable et de la Mer]

Technical staff

Danel Ahman [Inria, from Apr 2017 until Sep 2017]
Gergely Bana [Inria, until Jan 2017, granted by FP7 ERC CIRCUS project]
Victor Dumitrescu [Inria]
Guglielmo Fachini [Inria]
Natalia Kulatova [Inria, until Oct 2017]
Tomer Libal [Inria, until Jul 2017]
Marc Sylvestre [Inria]

Interns

Carmine Abate [Inria, from Dec 2017]
William Bowman [Inria, from Oct 2017]
Keith Cannon [Inria, from Mar 2017 until Sep 2017]
Theo Laurent [Inria, from Mar 2017 until Aug 2017]
Benjamin Lipp [Inria, from Dec 2017]
Clement Pit Claudel [Inria, from Jul 2017 until Oct 2017]

Administrative Assistants

Anna Bednarik [Inria]
Helene Milome [Inria]
Mathieu Mourey [Inria]

Visiting Scientists

Ana Evans [University of Virginia, from Apr 2017 until Aug 2017]
David Evans [University of Virginia, from Apr 2017 until Aug 2017]
Lucca Hirschi [Ministère de l'Enseignement Supérieur et de la Recherche, until Sep 2017]
Jake Silverman [Inria, from Jun 2017 until Aug 2017]
Aaron Weiss [Northeastern University, from Sep 2017]

External Collaborators

David Baelde [Ecole Normale Supérieure Cachan, from Sep 2017]
Theo Laurent [Ecole Normale Supérieure Paris, from Aug 2017]
Jonathan Protzenko [Microsoft Research]

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 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 manual proofs. Programs written in F* can be translated to OCaml, F#, or C for execution.

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 security-oriented programming language and type system F*. 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 computationally sound type system F* for applications written in F#. 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 compilers for realistic programming languages, both low-level (the C language) and high-level (ML and F*, a dependently-typed variant). These compilers will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilers will target a tagged architecture, 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.

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 2014) 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, the 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 dependent type system 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 [59]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [57] 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* [54]. ProVerif also distinguishes itself from other tools by the variety of cryptographic primitives it can handle, defined by rewrite rules or by some equations, and the variety of security properties it can prove: secrecy [52], [43], correspondences (including authentication) [53], and observational equivalences [51]. 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 is the only tool that proves equivalences for an unbounded number of sessions.

Using ProVerif, it is now possible to verify large parts of industrial-strength protocols, such as TLS [48], JFK [44], and Web Services Security [50], 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 30 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 [55], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [56]. 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, that models typically ignore. This leads to a situation that a protocol may have been proved secure in theory, but 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# [48]. 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 such as typechecking. F7 [46] is a refinement typechecker for F#, developed jointly at Microsoft Research Cambridge and Inria. It implements a dependent type-system that allows us to specify security assumptions and goals as first-order logic annotations directly inside the program. It has been used for the modular verification of large web services security protocol implementations [49]. F* (see below) is an extension of F7 with higher-order kinds and a certifying typechecker. The cryptographic protocol implementations verified using F7 and F* already represent the largest verified cryptographic applications to our knowledge.

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 already designed the automatic tool CryptoVerif, which generates proofs by sequences of games. Much work is still needed in order to develop this approach, so that it is applicable to more protocols. We also plan to design and implement techniques for proving implementations of protocols secure in the computational model, by generating them from CryptoVerif specifications that have been proved secure, or by automatically extracting CryptoVerif models from implementations.

A different approach is to directly verify cryptographic applications in the computational model by typing. A recent work [58] 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 Designed for Program Verification

F* [60] is a verification system for ML programs developed collaboratively by Inria and Microsoft Research. ML types are extended with logical predicates that can conveniently express precise specifications for programs (pre- and post- conditions of functions as well as stateful invariants), including functional correctness and security properties. The F* typechecker implements a weakest-precondition calculus to produce first-order logic formulas that are automatically discharged using the Z3 SMT solver. The original F* implementation has been successfully used to verify nearly 50,000 lines of code, including cryptographic protocol implementations, web browser extensions, cloudhosted web applications, and key parts of the F* typechecker and compiler (itself written in F*). F* has also been used for formalizing the semantics of other languages, including JavaScript and a compiler from a subset of F* to JavaScript, and TS*, a secure subset of TypeScript. Programs verified with F* can be extracted to F#, OCaml, C, and JavaScript and then efficiently executed and integrated into larger code bases.

The latest version of F* is written entirely in F*, and bootstraps in OCaml and F#. It is open source and under active development on GitHub. A detailed description of this new F* version is available in a POPL 2016 paper [62] and a POPL 2017 one [22]. We continue to evolve and develop F* and we use it to develop large case studies of verified cryptographic applications, such as miTLS.

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 compilers for realistic programming languages, both low-level (the C language) and high-level (ML and F*, a dependently-typed variant). These compilers will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilers will target a tagged architecture, which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We will experimentally evaluate and carefully optimize the efficiency of our secure compilers on realistic workloads and standard benchmark suites. We will use property-based testing and formal verification to provide high confidence that our compilers are indeed secure. Formally, we will construct machine-checked proofs of full abstraction with respect to a secure high-level semantics. This strong property complements compiler correctness and ensures that no machine-code attacker can do more harm to securely compiled components than a component in the secure source language already could.

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 [47] [47] and TS* [61], 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 the 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.

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 20 papers at top-tier conferences such as POPL (2), IEEE S&P (2), ACM CCS (1), IEEE CSF (1), ICFP (1), PETS (1), and IEEE Euro S&P (2).
- Bruno Blanchet published a paper on the applied pi calculus in the prestigious Journal of the ACM.
- The HACL* verified cryptographic library developed in our group was integrated into Mozilla Firefox 57 and is being actively used by hundreds of millions of users around the world.
- We organized the second edition of the IEEE Euro S&P Conference in Paris, which was attended by over 200 security researchers from around the world.

5.1.1. Awards

- Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi won a Distinguished Paper award at IEEE S&P 2017 .
- Catalin Hritcu was awarded a new DARPA SSITH grant called HOPE with DRAPER Labs.
- Antoine Delignat-Lavaud received an “accessit” for the prix de thèse GDR GPL 2016.

BEST PAPER AWARD:

[24]
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

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 made several case studies using CryptoVerif (Signal, TLS 1.3 Draft 18, ARINC 823 avionic protocol) and have made a few technical improvements.

- Participants: Bruno Blanchet and David Cadé
- Contact: Bruno Blanchet
- Publications: 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, Cédric 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, Cédric Fournet, Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Pierre-Yves Strub and Santiago Zanella-Béguelin
- 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: Marc Sylvestre improved the display of attacks, in particular by showing the computations performed by the attacker to obtain the messages sent in the attack, and by explaining why the found trace breaks the considered security property. He also developed an interactive simulator that allows the user to run the protocol step by step. We also made several case studies using this tool (Signal, TLS 1.3 Draft 18, ARINC 823 avionic protocol).

- 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 - 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 in the Symbolic Model

Participants: Bruno Blanchet, Marc Sylvestre.

The applied pi calculus is a widely used language for modeling security protocols, including as a theoretical basis of PROVERIF. However, the seminal paper that describes this language [45] does not come with proofs, and detailed proofs for the results in this paper were never published. Martín Abadi, Bruno Blanchet, and Cédric Fournet wrote detailed proofs of all results of this paper. This work appears in the Journal of the ACM [12].

Marc Sylvestre improved the display of attacks in ProVerif, in particular by showing the computations performed by the attacker to obtain the messages sent in the attack, and by explaining why the found trace breaks the considered security property. He also developed an interactive simulator that allows the user to run the protocol step by step. The extended tool is available at http://proverif.inria.fr.

7.2. Symbolic and Computational Verification of Signal

Participants: Karthikeyan Bhargavan, Bruno Blanchet, Nadim Kobeissi.

We proposed a novel methodology that allows protocol designers, implementers, and security analysts to collaboratively verify a protocol using automated tools. The protocol is implemented in ProScript, a new domain-specific language that is designed for writing cryptographic protocol code that can both be executed within JavaScript programs and automatically translated to a readable model in the applied pi calculus. This model can then be analyzed symbolically using ProVerif to find attacks in a variety of threat models. The model can also be used as the basis of a computational proof using CryptoVerif, which reduces the security of the protocol to standard cryptographic assumptions. If ProVerif finds an attack, or if the CryptoVerif proof reveals a weakness, the protocol designer modifies the ProScript protocol code and regenerates the model to enable a new analysis. We demonstrated our methodology by implementing and analyzing two protocols: a variant of the popular Signal Protocol and TLS 1.3 Draft-18.

In our analysis of Signal, we used ProVerif and CryptoVerif to find new and previously-known weaknesses in the protocol and suggest practical countermeasures. Our ProScript protocol code is incorporated within the current release of Cryptocat, a desktop secure messenger application written in JavaScript. Our results indicate that, with disciplined programming and some verification expertise, the systematic analysis of complex cryptographic web applications is now becoming practical [33].

7.3. Symbolic and Computational Verification of TLS 1.3

Participants: Karthikeyan Bhargavan, Bruno Blanchet, Nadim Kobeissi.

We also applied our verification methodology to TLS 1.3, the next version of the Transport Layer Security (TLS) protocol. Its clean-slate design is a reaction both to the increasing demand for low-latency HTTPS connections and to a series of recent high-profile attacks on TLS. The hope is that a fresh protocol with modern cryptography will prevent legacy problems; the danger is that it will expose new kinds of attacks, or reintroduce old flaws that were fixed in previous versions of TLS. The protocol is nearing completion, and the working group has appealed to researchers to analyze the protocol before publication. We responded by presenting a comprehensive analysis of the TLS 1.3 Draft-18 protocol.

We seeked to answer three questions that had not been fully addressed in previous work on TLS 1.3: (1) Does TLS 1.3 prevent well-known attacks on TLS 1.2, such as Logjam or the Triple Handshake, even if it is run in parallel with TLS 1.2? (2) Can we mechanically verify the computational security of TLS 1.3 under standard (strong) assumptions on its cryptographic primitives? (3) How can we extend the guarantees of the TLS 1.3 protocol to the details of its implementations?

To answer these questions, we used our methodology for developing verified symbolic and computational models of TLS 1.3 hand-in-hand with a high-assurance reference implementation of the protocol. We presented symbolic ProVerif models for various intermediate versions of TLS 1.3 and evaluated them against a rich class of attacks to reconstruct both known and previously unpublished vulnerabilities that influenced the current design of the protocol. We presented a computational CryptoVerif model for TLS 1.3 Draft-18 and proved its security. We presented RefTLS, an interoperable implementation of TLS 1.0-1.3 in ProScript and automatically analyzed its protocol core by extracting a ProVerif model from its typed JavaScript code [24], [37]. This work was awarded the Distinguished Paper award at IEEE S&P 2017.

7.4. Verification of Avionic Security Protocols

Participant: Bruno Blanchet.

Within the ANR project AnaStaSec, we studied an air-ground avionic security protocol, the ARINC823 public key protocol [41]. We verified this protocol both in the symbolic model of cryptography, using ProVerif, and in the computational model, using CryptoVerif. While this study confirmed the main security properties of the protocol (entity and message authentication, secrecy), we found several weaknesses and imprecisions in the standard. We proposed fixes for these problems. This work appears in [27], [38].

We also verified the ATN Secure Dialogue protocol (ICAO 9880-IV [42]), which is currently under development. We verified it using ProVerif and CryptoVerif. While we confirmed the main security properties of the intended protocol, we found several incoherences, weaknesses, and imprecisions in the draft standard. We proposed fixes for these problems. We presented this work to the ICAO Secure Dialogue Subgroup (September 2017).

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

Participants: Harry Halpin, George Danezis [University College London], Carmela Troncoso [IMDEA].

We continued work on next-generation protocols via the NEXTLEAP project in 2017. The work started in 2016 to define the principles of design of decentralized protocols and a paper was published in the Privacy Enhancing Techologies Symposium as "Systematizing Decentralization and Privacy: Lessons from 15 years of research and deployments", which systematized over 180 papers from p2p to blockchains. We formally defined decentralization in terms of a distributed system operating in an adversarial environment, which we hope will be a foundational contribution to the field. NEXTLEAP also published a paper in ARES 2017 on how these principles can be applied to secure messaging systems, including the work of Prosecco on formalizing secure messaging as presented in EuroS&P 2017. NEXTLEAP had a successful launch event at Centre Pompidou, colocated with Eurocrypt, which was attended by a panel of prominent cryptographers (Phil Rogaway, Moti Yung, Tanja Lange, Daniel Bernstein) and members of the European Commission and European Parliament, attracting over 100 members of the general public to hear about Prosecco's research.

Building on the work on identity started in 2017, we finished the design of ClaimChain, the privacy-enhanced blockchain-based identity system, and work started on a F* implementation and scalability simulations. Unlike most blockchain systems that are public and are essentially replicated state machines, Claimchains use VRFs for privacy and do not require global consensus, instead allowing private linking between Claimchains and gossiping to maintain local consensus on secret material. We believe that this design may be the first workable approach to decentralizing PKI. Claimchains also use Merkle Trees for efficiency, and some of this library may end up as generally useful for F* programming after more development in 2018. Claimchain has yet to be published in an academic venue, but it has already attracted considerable interest and was presented in the popular CCC security conference in Leipzig Germany. We also continued to raise the bar on security and privacy, hosting the first ever workshop on "Security and Privacy on the Blockchain" at EuroS&P 2017, which was sponsored by Blockstream. We expect the first formally verified blockchain system based on this design to be finished in 2018.

Another aspect of building next-generation protocols is to evaluate their usability. Prior studies have shown that users typically do not understand encryption and are even hostile to open-source code. However, these studies are typically done with students drawn for a general population, and in response Prosecco, in co-operation with sociologists from CNRS/Sorbonne, have started the largest-ever study of high-risk users from countries as diverse as Ukraine, Russia, Egypt and Tunisia. Preliminary results were presented at the European Usable Security (EuroUSEC) workshop, and already have attracted considerable attention from developers of secure messaging applications such as Signal and Briar. We hope that our findings on how users actually do group messaging and key verification will lead to changes in the underlying protocols.

Lastly, we continue to work with standards bodies in order to do security and privacy analysis of new protocols. For example, we have started formalizing W3C Web Authentication and inspecting its privacy properties, and our work on the lack of security in Semantic Web standards led to "Semantic Insecurity: Security and the Semantic Web" at ISWC 2017. Work on the security and privacy properties of the W3C Encrypted Media Extension led to an invited keynote at SPACE 2017.

Next year, we will finalize ClaimChain and add on the mix-network we have been developing over the last year, leading to a metadata-resistant and decentralized secure messaging application. We will work on spreading awareness of the importance of formally verified open standards as being necessary for the future of security, rather than closed-source solutions that may have backdoors and dangerous bugs that could cause severe economic damage if not fixed. To this end, we will work with ECRYPT CSA on the IACR Summer School of Societal and Business Impact of Cryptography, colocated with Real-World Crypto 2018, and co-organize an event at the European Commission and Parliament.

7.6. The F* programming language

Participants: Danel Ahman, Benjamin Beurdouche, Karthikeyan Bhargavan, Barry Bond [Microsoft Research], Tej Chajed [MIT], Antoine Delignat-Lavaud [Microsoft Research], Victor Dumitrescu, Cédric Fournet [Microsoft Research], Catalin Hritcu, Qunyan Mangus [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Kenji Maillard, Asher Manning [McGill University], Guido Martínez [CIFASIS-CONICET Rosario], Zoe Paraskevopoulou [Princeton University], Clément Pit-Claudel [MIT], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Jared Roesch [University of Washington], Nikhil Swamy [Microsoft Research], Christoph M. Wintersteiger [Microsoft Research], Santiago Zanella-Béguelin [Microsoft Research].

F* 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 manual proofs. Programs written in F* can be translated to OCaml, F#, or C for execution.

The latest version of F* is written entirely in F*, and bootstraps in OCaml and F#. It is open source and under active development on http://github.com/FStarLang/FStar. A detailed description of this new F* version is available in a series of POPL papers [62], [22], [14].

The main ongoing use case of F* is building a verified, drop-in replacement for the whole HTTPS stack in Project Everest [25]. This includes verified implementations of TLS 1.2 and 1.3 including the underlying cryptographic primitives. Moreover, while F* is extracted to OCaml by default, we have devised a subset of F* that can be compiled to C for efficiency [18].

We released two versions of the software this year.

7.7. Micro-Policies

Participants: Arthur Azevedo de Amorim [University of Pennsylvania], Chris Casinghino [Draper Labs], André Dehon [University of Pennsylvania], Catalin Hritcu, Théo Laurent [ENS Paris], Benjamin Pierce [University of Pennsylvania], Howard Shrobe [MIT], Greg Sullivan [Dover Microsystems], Andrew Tolmach [Portland State University].

This year we obtained a new DARPA grant called SSITH/HOPE on “Advanced New Hardware Optimized for Policy Enforcement, A New HOPE”. This grant is in the process of starting and our contribution will focus on devising a high-level micro-policy language and investigating micro-policies targetting today’s most severe security vulnerabilities.

7.8. HACL*: A Verified Modern Cryptographic Library

Participants: Jean Karim Zinzindohoue, Karthikeyan Bhargavan, Jonathan Protzenko [Microsoft Research], Benjamin Beurdouche.

HACL* is a verified portable C cryptographic library that implements modern cryptographic primitives such as the ChaCha20 and Salsa20 encryption algorithms, Poly1305 and HMAC message authentication, SHA-256 and SHA-512 hash functions, the Curve25519 elliptic curve, and Ed25519 signatures.

HACL* is written in the F* programming language and then compiled to readable C code using the KreMLin tool [18]. The F* source code for each cryptographic primitive is verified for memory safety, mitigations against timing side-channels, and functional correctness with respect to a succinct high-level specification of the primitive derived from its published standard. The translation from F* to C preserves these properties and the generated C code can itself be compiled via the CompCert verified C compiler or mainstream compilers like GCC or CLANG. When compiled with GCC on 64-bit platforms, our primitives are as fast as the fastest pure C implementations in OpenSSL and Libsodium, significantly faster than the reference C code in TweetNaCl, and between 1.1x-5.7x slower than the fastest hand-optimized vectorized assembly code in the SUPERCOP benchmark test-suite.

HACL* implements the NaCl cryptographic API and can be used as a drop-in replacement for NaCl libraries like Libsodium and TweetNaCl. HACL* provides the cryptographic components for a new mandatory ciphersuite in TLS 1.3 and is being developed as the main cryptographic provider for the miTLS verified implementation. Primitives from HACL* have now been integrated within Mozilla's NSS cryptographic library. Our results show that writing fast, verified, and usable C cryptographic libraries is now practical.

This work appeared at the ACM CCS conference [36] and all our software is publicly available and in active development on GitHub.

7.9. miTLS: A Verified TLS Implementation

Participants: Karthikeyan Bhargavan, Antoine Delignat-Lavaud [Microsoft Research], Cédric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Jianyang Pan, Jonathan Protzenko [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoue.

The record layer is the main bridge between TLS applications and internal sub-protocols. Its core functionality is an elaborate authenticated encryption: streams of messages for each sub-protocol (handshake, alert, and application data) are fragmented, multiplexed, and encrypted with optional padding to hide their lengths. Conversely, the sub-protocols may provide fresh keys or signal stream termination to the record layer.

Compared to prior versions, TLS 1.3 discards obsolete schemes in favor of a common construction for Authenticated Encryption with Associated Data (AEAD), instantiated with algorithms such as AES-GCM and ChaCha20-Poly1305. It differs from TLS 1.2 in its use of padding, associated data and nonces. It encrypts the content-type used to multiplex between sub-protocols. New protocol features such as early application data (0-RTT and 0.5-RTT) and late handshake messages require additional keys and a more general model of stateful encryption.

As part of the miTLS project, we built and verified a reference implementation of the TLS record layer and its cryptographic algorithms in F*. We reduced the high-level security of the record layer to cryptographic assumptions on its ciphers. Each step in the reduction is verified by typing an F* module; when the step incurs a security loss, this module precisely captures the corresponding game-based security assumption.

We computed concrete security bounds for the AES-GCM and ChaCha20-Poly1305 ciphersuites, and derived recommended limits on sent data before re-keying. Combining our functional correctness and security results, we obtained the first verified implementation of the main TLS 1.3 record ciphers. We plugged our implementation into an existing TLS library and confirmed that the combination interoperates with Chrome and Firefox, and thus that experimentally the new TLS record layer (as described in RFCs and cryptographic standards) is provably secure.

This work appeared at IEEE S&P 2017 [26] and our verified software is publicly available and actively developed on GitHub.

7.10. A Cryptographic Analysis of Content Delivery of TLS

Participants: Karthikeyan Bhargavan, Ioana Boureanu [University of Surrey], Pierre-Alain Fouque [University of Rennes 1/IRISA], Cristina Onete [University of Rennes 1/IRISA], Benjamin Richard [Orange Labs Chatillon].

The Transport Layer Security (TLS) protocol is designed to allow two parties, a client and a server, to communicate securely over an insecure network. However, when TLS connections are proxied through an intermediate middlebox, like a Content Delivery Network (CDN), the standard end-to-end security guarantees of the protocol no longer apply.

As part of the SafeTLS project, we investigated the security guarantees provided by Keyless SSL, a CDN architecture currently deployed by CloudFlare that composes two TLS 1.2 handshakes to obtain a proxied TLS connection. We demonstrated new attacks that show that Keyless SSL does not meet its intended security goals. We argued that proxied TLS handshakes require a new, stronger, 3-party security definition, and we presented one.

We modified Keyless SSL and proved that our modifications guarantee this notion of security. Notably, we showed that secure proxying in TLS 1.3 is computationally lighter and requires simpler assumptions on the certificate infrastructure than our proposed fix for Keyless SSL. Our results indicate that proxied TLS architectures, as currently used by a number of CDNs, may be vulnerable to subtle attacks and deserve close attention [39].

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. AnaStaSec

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

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

Duration: January 2015 - December 2018.

Coordinator: Jérôme Féret, Inria Antique (France)

Participant: Bruno Blanchet

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

8.1.1.2. 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.

8.1.1.3. 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, Univesité 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.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. 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 Bhargavn, Inria

Abstract: 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.

8.2.1.2. ERC Starting Grant: SECOMP

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

Duration: Jan 2017 - December 2021

Coordinator: Catalin Hritcu, Inria

Abstract: This new ERC-funded project called SECOMP1 is aimed at leveraging emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilers for realistic programming languages, both low-level (the C language) and high-level (F*, a dependently-typed ML variant). These compilers will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilers will target a tagged architecture, 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.

8.2.1.3. NEXTLEAP

Title: NEXTLEAP: NEXT generation Legal Encryption And Privacy

Programme: H2020

Duration: January 2016 - December 2018

Coordinator: Harry Halpin, Inria

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

Abstract: NEXTLEAP aims to create, validate, and deploy protocols that can serve as pillars for a secure, trust-worthy, and privacy-respecting Internet. For this purpose NEXTLEAP will develop an interdisciplinary study of decentralisation that provides the basis on which these protocols can be designed, working with sociologists to understand user needs. The modular specification of decentralized protocols, implemented as verified open-source software modules, will be done for both privacy-preserving secure federated identity as well as decentralized secure messaging services that hide metadata (e.g., who, when, how often, etc.).

8.3. International Initiatives

8.3.1. Inria International Labs

8.3.1.1. Informal International Partners

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

- **F***: Microsoft Research (Cambdridge, Redmond), IMDEA (Madrid)
- **TLS analysis**: Microsoft Research (Cambridge), Mozilla, University of Rennes
- **Web Security**: Microsoft Research (Cambridge, Redmond), Imperial College (London), University of Stuttgart
- **Micro-Policies**: University of Pennsylvania, Portland State University

8.3.2. Participation in Other International Programs

8.3.2.1. International Initiatives

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

Program: DARPA SSITH

Duration: January 2016 - December 2018

Coordinator: Charles Stark, Draper Laboratory

Participants: Catalin Hritcu

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.

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Claudia Diaz from KUL visited the group from 1-2 March and gave a seminar "Designing Mix-nets"
- Peter Schwabe visited Inria Paris on 11 April; he gave a seminar: From NewHope to Kyber.
- Joseph Bonneau (Stanford University) visited Inria on 20 April 2017, he gave a seminar: Public randomness, blockchains and proofs-of-delay
- Stefan Ciobaca (Alexandru Ioan Cuza University of Iai, Romania) visited Inria Paris on 15 May 2017; he gave a seminar: The RMT Tool for Rewriting Modulo Theories.
- Ana Nora Evans (University of Virginia) joined Inria as a Visiting Scientist Apr–Aug 2017; she gave a seminar: Using Verified Software Fault Isolation for a Formally Secure Compiler.
- David Evans (University of Virginia) joined Inria as a Visiting Scientist Apr–Aug 2017; he gave a seminar: Can Machine Learning Work in the Presence of Adversaries?
- Jean Yang (CMU) visited Inria Paris on 6 June 2017; she gave a seminar: Policy-Agnostic Programming for Database-Backed Applications.
- Amal Ahmed (Northeastern University) joined Inria as a Visiting Professor from September 2017; she gave a seminar: Prosecco Seminars: Compositional Compiler Verification for a Multi-Language World.
- Aaron Weiss (Northeastern University) joined Inria as a Visiting Scientist from September 2017.
- Amin Timany (KU Leuven) visited Inria Paris 6-8 December 2017; he gave a seminar: A Logical Relation for Monadic Encapsulation of State: Proving contextual equivalences in the presence of runST.
- Eric Rescorla visited Prosecco to discuss the design of TLS 1.3.

8.4.1.1. Internships

- Benjamin Lipp: Dec 2017 until May 2018, supervised by B. Blanchet, K. Bhargavan, and H. Halpin
- Iness Ben Guirat: Masters student 2017, supervised by H. Halpin
- Carmine Abate (University of Trento): Dec 2017 until May 2018
- William Bowman (Northeastern University): Oct 2017 until Dec 2017
- Keith Cannon (American University Paris): Mar 2017 until Sep 2017
- Théo Laurent (ENS Paris): Mar 2017 until Aug 2017
- Clément Pit-Claudel (MIT): Jul 2017 until Oct 2017

8.4.2. Visits to International Teams

- Catalin Hritcu, October 8-13, 2017, Aarhus University, Denmark.
- Catalin Hritcu, October 16-17, 2017, MPI-SWS, Saarbrucken, Germany.

- Catalin Hritcu, December 18, 2017, University of Iasi, Romania.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

- Prosecco organized the 2nd IEEE European Symposium on Security and Privacy in Paris, 26-28 April 2017. Catalin Hritcu was General Chair, Bruno Blanchet was Finance Chair, and Karthikeyan Bhargavan was Local arrangements Chair.
- Harry Halpin co-chaired the IEEE Security and Privacy on the Blockchain workshop, colocated with IEEE EuroS&P, on 29 April 2017.
- Catalin Hritcu is Artifact Evaluation Co-Chair of POPL 2018
- Catalin Hritcu created a New Workshop on Principles of Secure Compilation (PriSC) colocated with POPL 2017 and 2018. He is PC Chair for PriSC 2018.
- Prosecco organized a Project Everest Workshop at Inria Paris, 2 October 2017
- Prosecco organized an ESOP PC workshop at Inria Paris, 15 December 2017 Workshop at POPL: 13 January 2018, Los Angeles, USA

9.1.2. Scientific Events Selection

9.1.2.1. Member of the Conference Program Committees

- Bruno Blanchet was PC member at TAP 2017.
- Harry Halpin was a PC member for ISWC 2017 and WWW 2017.
- Catalin Hritcu was PC member at ESOP 2018 and EuroS&P 2018
- Karthikeyan Bhargavan was a PC member at ACM CCS 2017-18, IEEE S&P 2017-18, POST 2018.

9.1.2.2. Reviewer

- Harry Halpin served as a reviewer for LatinCrypt, AsiaCrypt, JAIST, TCS

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

Associate Editor

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

9.1.4. Invited Talks

- Bruno Blanchet gave an invited talk at the workshop on Models and Tools for Security Analysis and Proofs, 2017.
- Bruno Blanchet gave an invited talk at the workshop TLS:DIV (TLS 1.3: Design, Implementation & Verification), 2017.
- Bruno Blanchet gave an invited talk at the workshop TMSP (Trends in Mechanized Security Proofs), 2017.
- Bruno Blanchet gave an invited talk at the Summer Research Institute, EPFL, 2017.
- Harry Halpin gave an invited talk at SPACE 2017
- Harry Halpin gave an invited talk at Conference on Privacy and Data Protection, January 2017.
- Harry Halpin gave an invited talk at RightsCon, March 2017.

- Harry Halpin gave an invited talk at E-CRYPT Cryptosympsium, March 2017.
- Harry Halpin gave an invited talk at La Firma Digital, July 2017.
- Harry Halpin gave an invited talk at Google, October 2017.
- Harry Halpin gave an invited talk at IMMWorld, November 2017.
- Harry Halpin gave an invited talk at Boston University Law School, November 2017.
- Harry Halpin gave an invited talk at University of North Carolina-Chapel Hill, December 2017.
- Harry Halpin gave a keynote talk at Security, and Privacy, and Cryptographic Engineering, December 2017.
- Catalin Hritcu was an invited speaker at TFP 2017
- Catalin Hritcu gave talks at Infoiasi, ESOP PC Workshop, Everest Workshop, TFP (Keynote), FADEx 2017, EuroS&P 2017, Université Clermont Auvergne, University Paris-Sud.
- Karthikeyan Bhargavan gave a keynote at ACNS 2017, Kanazawa, Japan.
- Karthikeyan Bhargavan gave an invited talk at Apple, Cupertino, USA.

9.1.5. 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.
- Karthikeyan Bhargavan advises the TLS working group at the IETF and consults for Mozilla, Apple, and Microsoft Research.
- Catalin Hritcu consilts for Microsoft Research and the DARPA SSITH/HOPE grant.

9.1.6. Research Administration

- Bruno Blanchet is a member of the Inria hiring committee for PhD, post-docs, and *délégations* (*Commision des Emplois Scientifiques*, CES).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Master: Catalin Hritcu, Cryptographic protocols: formal and computational proofs, 31.5h equivalent TD, master M2 MPRI, université Paris VII, France
- Doctorat: Catalin Hritcu: Verifying Cryptographic Implementations with F* at Computer-aided security proofs summer school. Aarhus, Denmark, October, 2017
- Doctorat: Catalin Hritcu: Verifying Cryptographic Implementations with F* course at Models and Tools for Cryptographic Proofs summer school, Nancy, France, July 2017
- Master: Karthikeyan Bhargavan, Cryptographic protocols: formal and computational proofs, 31.5h equivalent TD, master M2 MPRI, université Paris VII, France
- Master: Karthikeyan Bhargavan, Protocol Verification and Safety, 18h equivalent TD, master ACN, Ecole Polytechnique et Telecom ParisTech, France

9.2.2. Supervision

- PhD: Evmorfia-Iro Bartzia, *A formalization of elliptic curves for cryptography*, Université Paris-Saclay, February 2017. Co-supervised by Pierre-Yves Strub and Karthikeyan Bhargavan.
- PhD in progress: Kenji Maillard, *Semantic Foundations for F**, started January 2017, supervised by Catalin Hritcu and Karthikeyan Bhargavan
- PhD in progress: Jean Karim Zinzindohoue, *A Verified Cryptographic Libary*, supervised by Karthikeyan Bhargavan

- PhD in progress: Nadim Kobeissi, 2015-, *Verified Web Security Applicaitons*, supervised by Karthikeyan Bhargavan
- PhD in progress: Benjamin Beurdouche, 2016-, *Verified Cryptographic Protocols for the Internet of Things*, supervised by Karthikeyan Bhargavan
- PhD in progress: Natalia Kulatova, 2017-, *Verified Hardware Security Devices*, co-supervised by Karthikeyan Bhargavan and Graham Steel
- PhD in progress: Marina Polybelova, 2017-, *Verified Cryptographic Web Applications*, supervised by Karthikeyan Bhargavan
- PhD in progress: Yaëlle Vincent, 2017-, *Software Security: combining fuzzing and symbolic execution for vulnerability detection*, co-supervised by Karthikeyan Bhargavan and Sebastien Bardin

9.2.3. Juries

- Bruno Blanchet was reviewer of Lucca Hirschi's PhD thesis.
- Harry Halpin served on the PhD jury of Evo Busseniers (Vrije Universitat Bruxelles)

9.3. Popularization

- Karthikeyan Bhargavan, Benjamin Beurdouche, Jean Karim Zinzindohoue published a paper in the Communications of the ACM.

10. 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° 1, pp. 1 - 103 [*DOI :* 10.1145/3127586], https://hal.inria.fr/hal-01636616

[2] 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)", Paris, France, ACM, 2017, pp. 515-529, https://arxiv.org/abs/1608.06499 [*DOI :* 10.1145/3009837.3009878], https://hal.archives-ouvertes.fr/hal-01424794

[3] R. BARDOU, R. FOCARDI, Y. KAWAMOTO, L. SIMIONATO, G. STEEL, J.-K. TSAY. *Efficient Padding Oracle Attacks on Cryptographic Hardware*, in "CRYPTO", 2012, pp. 608–625

[4] 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

[5] 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

[6] B. BLANCHET. *A Computationally Sound Mechanized Prover for Security Protocols*, in "IEEE Transactions on Dependable and Secure Computing", 2008, vol. 5, n° 4, pp. 193–207, Special issue IEEE Symposium on Security and Privacy 2006

[7] 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

[8] C. HRITCU, M. GREENBERG, B. KAREL, B. C. PIERCE, G. MORRISETT. *All Your IFCException Are Belong to Us*, in "IEEE Symposium on Security and Privacy (Oakland)", 2013, pp. 3–17

[9] 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, http://doi.acm.org/10.1145/2994620.2994637

[10] Y. JUGLARET, C. HRITCU, A. AZEVEDO DE AMORIM, B. ENG, B. C. PIERCE. *Beyond Good and Evil: Formalizing the Security Guarantees of Compartmentalizing Compilation*, in "29th IEEE Symposium on Computer Security Foundations (CSF)", IEEE Computer Society Press, July 2016, pp. 45–60 [*DOI :* 10.1109/CSF.2016.11], http://arxiv.org/abs/1602.04503

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] E.-I. BARTZIA. *A formalization of elliptic curves for cryptography*, Université Paris-Saclay, February 2017, https://pastel.archives-ouvertes.fr/tel-01563979

Articles in International Peer-Reviewed Journals

[12] 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

[13] D. AHMAN. *Handling Fibred Algebraic Effects*, in "Proceedings of the ACM on Programming Languages", January 2018, vol. 2, n^{o} POPL [*DOI :* 10.1145/3158095], https://hal.archives-ouvertes.fr/hal-01672734

[14] D. AHMAN, C. FOURNET, C. HRIŢCU, K. MAILLARD, A. RASTOGI, N. SWAMY. *Recalling a Witness: Foundations and Applications of Monotonic State*, in "Proceedings of the ACM on Programming Languages", January 2018, vol. 2, n^{o} POPL, https://arxiv.org/abs/1707.02466 [*DOI :* 10.1145/3158153], https://hal.archives-ouvertes.fr/hal-01672733

[15] K. BHARGAVAN, B. BEURDOUCHE, A. DELIGNAT-LAVAUD, C. FOURNET, M. KOHLWEISS, A. PIRONTI, P.-Y. STRUB, J. K. ZINZINDOHOUE. *A messy state of the union*, in "Communications of the ACM", January 2017, vol. 60, n^{o} 2, pp. 99 - 107 [*DOI :* 10.1145/3023357], https://hal.inria.fr/hal-01673714

[16] W. J. BOWMAN, Y. CONG, N. RIOUX, A. AHMED. *Type-Preserving CPS Translation of* Σ *and* Π *Types is Not Not Possible*, in "Proceedings of the ACM on Programming Languages", January 2018, vol. 2, n^{o} POPL [*DOI :* 10.1145/3158110], https://hal.archives-ouvertes.fr/hal-01672735

[17] O. FLÜCKIGER, G. SCHERER, M.-H. YEE, A. GOEL, A. AHMED, J. VITEK. *Correctness of Speculative Optimizations with Dynamic Deoptimization*, in "Proceedings of the ACM on Programming Languages", 2017, https://arxiv.org/abs/1711.03050 , forthcoming [*DOI :* 10.1145/3158137], https://hal.inria.fr/hal-01646765

[18] J. PROTZENKO, J. 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 "Proceedings of the ACM on Programming Languages", September 2017, vol. 1, n° ICFP, pp. 17:1–17:29, https://arxiv.org/abs/1703.00053 [*DOI :* 10.1145/3110261], https://hal.archives-ouvertes.fr/hal-01672706

[19] C. TRONCOSO, M. ISAAKIDIS, G. DANEZIS, H. HALPIN. *Systematizing Decentralization and Privacy: Lessons from 15 Years of Research and Deployments*, in "Proceedings on Privacy Enhancing Technologies", October 2017, vol. 2017, n° 4, pp. 307 - 329 [*DOI :* 10.1515/POPETS-2017-0056], https://hal.inria.fr/hal-01673295

Invited Conferences

[20] H. HALPIN. *The Crisis of Standardizing DRM: The Case of W3C Encrypted Media Extensions*, in "SPACE 2017 - Seventh International Conference on Security, Privacy, and Applied Cryptography Engineering", Goa, India, Lecture Notes in Computer Science, Springer, December 2017, vol. 10662, pp. 10-29 [*DOI :* 10.1007/978-3-319-71501-8_2], https://hal.inria.fr/hal-01673296

[21] G. LEURENT, K. BHARGAVAN. *On the Practical (In-)Security of 64-bit Block Ciphers*, in "ESC 2017 - Early Symmetric Crypto", Canach, Luxembourg, January 2017, https://hal.inria.fr/hal-01105128

International Conferences with Proceedings

[22] 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)", Paris, France, ACM, 2017, pp. 515-529, https://arxiv.org/abs/1608.06499 [*DOI :* 10.1145/3009837.3009878], https://hal.archives-ouvertes.fr/hal-01424794

[23] D. AHMAN, T. UUSTALU. *Taking Updates Seriously*, in "Proceedings of the 6th International Workshop on Bidirectional Transformations co-located with The European Joint Conferences on Theory and Practice of Software - ETAPS 2017", Uppsala, Sweden, April 2017, pp. 59–73, https://hal.archives-ouvertes.fr/hal-01672736

[24] *Best Paper*
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.

[25] K. BHARGAVAN, B. BOND, A. DELIGNAT-LAVAUD, C. FOURNET, C. HAWBLITZEL, C. HRITCU, S. ISHTIAQ, M. KOHLWEISS, R. LEINO, J. LORCH, K. MAILLARD, J. PAIN, 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)", Asilomar, CA, United States, May 2017 [*DOI :* 10.4230/LIPICS.SNAPL.2017.1], https://hal.archives-ouvertes.fr/hal-01672707

[26] 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 "SP 2017 - 38th IEEE Symposium on Security and Privacy", San Jose, United States, May 2017, pp. 463-482 [*DOI :* 10.1109/SP.2017.58], https://hal.inria.fr/hal-01674096

[27] B. BLANCHET. *Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols*, in "30th IEEE Computer Security Foundations Symposium", Santa Barbara, United States, August 2017, pp. 68-82 [*DOI :* 10.1109/CSF.2017.7], https://hal.inria.fr/hal-01575861

[28] K. CAIRNS, H. HALPIN, G. STEEL. *Security Analysis of the W3C Web Cryptography API*, in "Proceedings of Security Standardisation Research (SSR)", Gaithersberg, United States, Lecture Notes in Computer Science (LNCS), Springer, December 2017, vol. 10074, pp. 112 - 140 [*DOI :* 10.1007/978-3-319-49100-4_5], https://hal.inria.fr/hal-01426852

[29] N. GRIMM, K. MAILLARD, C. FOURNET, C. HRIŢCU, M. MAFFEI, J. PROTZENKO, T. RAMANANANDRO, A. RASTOGI, N. SWAMY, S. ZANELLA-BÉGUELIN. *A Monadic Framework for Relational Verification: Applied to Information Security, Program Equivalence, and Optimizations*, in "7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)", Los Angeles, United States, ACM, January 2018, pp. 130–145, https://arxiv.org/abs/1703.00055 [*DOI :* 10.1145/3167090], https://hal.archives-ouvertes.fr/hal-01672703

[30] H. HALPIN. *A Roadmap for High Assurance Cryptography*, in "FPS 2017 - 10th International Symposium on Foundations & Practice of Security", Nancy, France, October 2017, pp. 1-9, https://hal.inria.fr/hal-01673294

[31] H. HALPIN. *NEXTLEAP: Decentralizing Identity with Privacy for Secure Messaging*, in "ARES 2017 - 12th International Conference on Availability, Reliability and Security", Reggio Calabria, Italy, ACM, August 2017, pp. 1-10 [*DOI :* 10.1145/3098954.3104056], https://hal.inria.fr/hal-01673292

[32] H. HALPIN. *Semantic Insecurity: Security and the Semantic Web*, in "Society, Privacy and the Semantic Web - Policy and Technology (PrivOn 2017)", Vienna, Austria, October 2017, https://hal.inria.fr/hal-01673291

[33] 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

[34] N. KOBEISSI, K. BHARGAVAN, B. BLANCHET. *Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach*, in "EuroS&P 2017 - 2nd IEEE European Symposium on Security and Privacy", Paris, France, A. SABELFELD, M. SMITH (editors), IEEE, April 2017, pp. 435 - 450 [*DOI :* 10.1109/EUROSP.2017.38], https://hal.inria.fr/hal-01583009

[35] L. LAMPROPOULOS, D. GALLOIS-WONG, C. HRIŢCU, J. HUGHES, B. C. PIERCE, L.-Y. XIA. *Beginner's Luck: A Language for Random Generators*, in "44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)", Paris, France, ACM, 2017, pp. 114-129, https://arxiv.org/abs/1607.05443 [*DOI :* 10.1145/3009837.3009868], https://hal.archives-ouvertes.fr/hal-01424793

[36] J.-K. ZINZINDOHOUÉ, K. BHARGAVAN, J. PROTZENKO, B. BEURDOUCHE. *HACL * : A Verified Modern Cryptographic Library*, in "ACM Conference on Computer and Communications Security (CCS)", Dallas, United States, October 2017, https://hal.inria.fr/hal-01588421

Research Reports

[37] K. BHARGAVAN, B. BLANCHET, N. KOBEISSI. *Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate*, Inria Paris, May 2017, n° RR-9040, 51 p. , https://hal.inria.fr/hal-01528752

[38] B. BLANCHET. *Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols*, Inria Paris, May 2017, n° RR-9072, 40 p. , https://hal.inria.fr/hal-01527671

Other Publications

[39] K. BHARGAVAN, I. BOUREANU, C. ONETE, P.-A. FOUQUE, B. RICHARD. *Content delivery over TLS: a cryptographic analysis of keyless SSL*, IEEE, April 2017, pp. 600-615, EuroS&P 2017 - 2nd IEEE European Symposium on Security and Privacy [*DOI :* 10.1109/EUROSP.2017.52], https://hal.inria.fr/hal-01673853

[40] H. HALPIN, M. PIEKARSKA. *Introduction to Security and Privacy on the Blockchain*, April 2017, 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&P Workshops 2017), https://hal.inria.fr/hal-01673293

References in notes

[41] *ARINC SPECIFICATION 823P1: DATALINK SECURITY, PART 1 â ACARS MESSAGE SECURITY*, December 2007

[42] *ICAO Doc 9880: Manual on Detailed Technical Specifications for the Aeronautical Telecommunication Network (ATN) using ISO/OSI Standards and Protocols, Part IV B — Security Services, Third edition (Proposed Draft)*, May 2017

[43] 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

[44] 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

[45] M. ABADI, C. FOURNET. *Mobile Values, New Names, and Secure Communication*, in "28th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)", London, United Kingdom, ACM Press, January 2001, pp. 104–115

[46] J. BENGTSON, K. BHARGAVAN, C. FOURNET, A. D. GORDON, S. MAFFEIS. *Refinement types for secure implementations*, in "ACM Trans. Program. Lang. Syst.", 2011, vol. 33, n° 2, 8 p.

[47] K. BHARGAVAN, A. DELIGNAT-LAVAUD, S. MAFFEIS. *Language-Based Defenses Against Untrusted Browser Origins*, in "Proceedings of the 22th USENIX Security Symposium", 2013

[48] K. BHARGAVAN, C. FOURNET, R. CORIN, E. ZALINESCU. *Verified Cryptographic Implementations for TLS*, in "ACM Transactions Inf. Syst. Secur.", March 2012, vol. 15, n° 1, 3:1 p.

[49] K. BHARGAVAN, C. FOURNET, A. D. GORDON. *Modular Verification of Security Protocol Code by Typing*, in "ACM Symposium on Principles of Programming Languages (POPL'10)", 2010, pp. 445–456

[50] K. BHARGAVAN, C. FOURNET, A. D. GORDON, N. SWAMY. *Verified Implementations of the Information Card Federated Identity-Management Protocol*, in "Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)", ACM Press, 2008, pp. 123–135

[51] 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

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

[53] B. BLANCHET. *Automatic Verification of Correspondences for Security Protocols*, in "Journal of Computer Security", July 2009, vol. 17, n^{o} 4, pp. 363–434

[54] B. BLANCHET, A. PODELSKI. *Verification of Cryptographic Protocols: Tagging Enforces Termination*, in "Theoretical Computer Science", March 2005, vol. 333, n^{o} 1-2, pp. 67–90, Special issue FoSSaCS'03

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

[56] S. DELAUNE, S. KREMER, G. STEEL. *Formal Analysis of PKCS#11 and Proprietary Extensions*, in "Journal of Computer Security", November 2010, vol. 18, n^{o} 6, pp. 1211-1245

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

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

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

[60] N. SWAMY, J. CHEN, C. FOURNET, P.-Y. STRUB, K. BHARGAVAN, J. YANG. *Secure distributed programming with value-dependent types*, in "16th ACM SIGPLAN international conference on Functional Programming", 2011, pp. 266-278

[61] 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

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