Activity Report 2016

Project-Team PROSECCO

Programming securely with cryptography

RESEARCH CENTER
Paris

THEME
Security and Confidentiality

Table of contents

1. Members **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 3
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 F7 and F* 5
3.2. Computational verification of cryptographic applications 5
3.3. F*: A Higher-Order Effectful Language Designed 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 7
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 **8**
6. New Software and Platforms **9**
6.1. ProVerif 9
6.2. CryptoVerif 9
6.3. miTLS 10
6.4. F* 10
6.5. HACL* 10
6.6. ProScript 11
6.7. QuickChick 11
6.8. Luck 11
6.9. Privacy-preserving federated identity 11
7. New Results **12**
7.1. Verification of Security Protocols in the Symbolic Model 12
7.2. Verification of Security Protocols in the Computational model 12
7.3. Verification of Avionic Security Protocols 12
7.4. The F* programming language 12
7.5. Dependable Property-Based Testing 13
7.6. Micro-Policies and Secure Compilation 13
7.7. miTLS: Proofs for TLS 1.3 13
7.8. Attacks on obsolete cryptography 14
7.9. HACL*: Verified cryptographic library 14
7.10. Design and Verification of next-generation protocols: identity, blockchains, and messaging 14
8. Partnerships and Cooperations **15**
8.1. National Initiatives 15
8.1.1.1. AnaStaSec 15
8.1.1.2. AJACS 16
8.1.1.3. SafeTLS 16
8.1.1.4. QuickChick 16

8.2. European Initiatives 16
8.2.1.1. ERC Consolidator Grant: CIRCUS 16
8.2.1.2. ERC Starting Grant: SECOMP 17
8.2.1.3. NEXTLEAP 17
8.3. International Initiatives 17
8.4. International Research Visitors 18
8.4.1. Visits of International Scientists 18
8.4.2. Visits to International Teams 18
9. Dissemination **18**
9.1. Promoting Scientific Activities 18
9.1.1. Scientific Events Organisation 18
9.1.2. Scientific Events Selection 18
9.1.3. Journal 19
9.1.4. Invited Talks 19
9.2. Teaching - Supervision - Juries 19
9.2.1. Teaching 19
9.2.2. Supervision 19
9.2.3. Juries 20
9.3. Popularization 20
10. Bibliography **20**

Project-Team PROSECCO

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

Keywords:

Computer Science and Digital Science:

1.1. - Architectures
1.1.8. - Security of architectures
1.2. - Networks
1.2.8. - Network security
1.3. - Distributed Systems
2. - Software
2.1. - Programming Languages
2.1.1. - Semantics of programming languages
2.1.3. - Functional programming
2.1.7. - Distributed programming
2.1.11. - Proof languages
2.2. - Compilation
2.2.1. - Static analysis
2.2.3. - Run-time systems
2.4. - Verification, reliability, certification
2.4.2. - Model-checking
2.4.3. - Proofs
2.5. - Software engineering
4. - Security and privacy
4.3. - Cryptography
4.3.3. - Cryptographic protocols
4.5. - Formal methods for security
4.6. - Authentication
4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

6. - IT and telecom
6.1. - Software industry
6.1.1. - Software engineering
6.3. - Network functions
6.3.1. - Web
6.3.2. - Network protocols
6.4. - Internet of things
9. - Society and Knowledge
9.8. - Privacy

1. Members

Research Scientists

Karthikeyan Bhargavan [Team leader, Inria, Senior Researcher, HDR]
Bruno Blanchet [Inria, Senior Researcher, HDR]
Harry Halpin [Inria, Starting Research position]
Catalin Hritcu [Inria, Researcher]
David Baelde [ENS Cachan, Délégation à Inria, Researcher]

Engineers

Gergely Bana [Inria, granted by FP7 ERC CIRCUS project]
Natalia Kulatova [Inria, from May 2016]
Marc Sylvestre [Inria, from Oct 2016]

PhD Students

Benjamin Beurdouche [Inria, granted by FP7 ERC CIRCUS project]
Yannis Juglaret [Inria, until Sep 2016]
Nadim Kobeissi [Inria]
Jean Karim Zinzindohoué [Min. Ecologie]

Visiting Scientist

Jonathan Protzenko [Microsoft Research, Oct 2016]

Administrative Assistant

Anna Bednarik [Inria]

Others

Alejandro Aguirre [Inria, from Apr 2016 until Aug 2016]
Abhishek Bichhawat [Inria, from Sep 2016 until Dec 2016]
Diane Gallois-Wong [ENS Paris, until Aug 2016]
Ritobroto Maitra [Inria, from May 2016 until Aug 2016]
Guido Martinez [Inria, until Jun 2016]
Jianyang Pan [Inria, from May 2016 until Aug 2016]
Marina Polubelova [Inria, from Sep 2016 until Nov 2016]
Vinay Yogendra [Inria, from May 2016 until Jul 2016]

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 type systems F7 and F* for the programming language 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
- F7: a security typechecker for cryptographic applications written in F#

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 [42]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [40] 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* [36]. 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 [34], [25], correspondences (including authentication) [35], and observational equivalences [33]. 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 [30], JFK [26], and Web Services Security [32], 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 [37], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [38]. 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 F7 and 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# [30]. 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 [28] 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 [31]. F* (see below) is an extension of F7 with higher-order kinds and a certifying typechecker. Both F7 and F* have a growing user community. 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 [41] 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* [43] 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 [20] and a POPL 2017 one [6]. 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 [29] [29] and TS* [45], 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

This year, we published 18 articles in international peer-reviewed journals and conferences, including papers in prestigious conferences such as POPL, IEEE S&P Oakland, ACM CCS, NDSS, CSF, and WPES. Notably, Bruno Blanchet published a book surveying the use of ProVerif, his state-of-the-art protocol verification tool. We also won several research awards for our work, detailed below. Our work also exposed two new attacks, SLOTH and SWEET32, on Transport Layer Security, resulting in security updates and CVEs in popular web browsers and VPN software.

5.1.1. Awards

- Catalin Hritcu was awarded an ERC Starting Grant
- Catalin Hritcu was awarded an ANR Jeune Chercheur/Jeune Chercheuse Grant
- Karthikeyan Bhargavan was awarded an ERC Consolidator Grant
- Karthikeyan Bhargavan and Gaëtan Leurent won a Best Paper award at NDSS 2016
- Karthikeyan Bhargavan, Cedric Fournet, Markulf Kohlweiss, and Alfredo Pironti were awarded the Levchin prize for contributions to Real-World Cryptography
- Karthikeyan Bhargavan was awarded a Microsoft Outstanding Collaborator Award
- Karthikeyan Bhargavan was awarded the Prix Inria – Académie des sciences du Jeune chercheur

6. New Software and Platforms

6.1. ProVerif

Participants: Bruno Blanchet [correspondant], Xavier Allamigeon [April–July 2004], Vincent Cheval [Sept. 2011–Dec. 2014], Benjamin Smyth [Sept. 2009–Feb. 2010], Marc Sylvestre [Oct. 2016–].

PROVERIF (http://proverif.inria.fr) 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 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.

The PROVERIF verifier can prove the following properties:

- secrecy (the adversary cannot obtain the secret);
- authentication and more generally correspondence properties, of the form "if an event has been executed, then other events have been executed as well";
- strong secrecy (the adversary does not see the difference when the value of the secret changes);
- equivalences between processes that differ only by terms.

PROVERIF is widely used by the research community on the verification of security protocols (see http://proverif.inria.fr/proverif-users.html for references).

PROVERIF is freely available on the web, at http://proverif.inria.fr, under the GPL license.

6.2. CryptoVerif

Participants: Bruno Blanchet [correspondant], David Cadé [Sept. 2009–Dec. 2013].

CRYPTOVERIF(http://cryptoverif.inria.fr) 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.

The generated proofs are proofs by sequences of games, as used by cryptographers. These proofs are valid for a number of sessions polynomial in the security parameter, in the presence of an active adversary. CRYPTOVERIF can also evaluate the probability of success of an attack against the protocol as a function of the probability of breaking each cryptographic primitive and of the number of sessions (exact security).

CRYPTOVERIF has been used in particular for a study of Kerberos in the computational model, and as a back-end for verifying implementations of protocols in F# and C.

CRYPTOVERIF is freely available on the web, at http://cryptoverif.inria.fr, under the CeCILL license.

6.3. miTLS

Participants: Karthikeyan Bhargavan [correspondant], Cedric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Antoine Delignat-Lavaud [Microsoft Research], Nikhil Swamy [Microsoft Research], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoué, Benjamin Beurdouche, Alfredo Pironti.

miTLS is a verified reference implementation of the TLS security protocol in F#, a dialect of OCaml for the .NET platform. It supports SSL version 3.0 and TLS versions 1.0-1.2 and interoperates with mainstream web browsers and servers. miTLS has been verified for functional correctness and cryptographic security using the refinement typechecker F7.

Papers describing the miTLS library was published at IEEE S&P 2013, CRYPTO 2014, and IEEE S&P Journal 2016. miTLS is now being developed on GitHub with dozens of contributors and regular updates. The miTLS team was awarded the Levchin prize for contributions to Real-World Cryptography in 2016. The software and associated research materials are available from http://mitls.org.

6.4. F*

Participants: Alejandro Aguirre, Danel Ahman [University of Edinburgh], Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud [Microsoft Research], Cédric Fournet [Microsoft Research], Catalin Hritcu, Chantal Keller [Université Paris-Sud], Kenji Maillard, Guido Martínez, Gordon Plotkin, Samin Ishtiaq [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research], Peng Wang [MIT], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoué.

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 Fω (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.

A detailed description of F* (circa 2011) appeared in the Journal of Functional Programming [44]. F* has evolved substantially since then. The latest version of F* is written entirely in F*, and bootstraps in OCaml and F#. It is under active development at GitHub: https://github.com/FStarLang and the official webpage is at http://fstar-lang.org.

6.5. HACL*

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

HACL* is a verified cryptographic library written in F*. It implements modern primitives, including elliptic curves like Curve25519, symmetric ciphers like Chacha20, and MAC algorithms like Poly1305. These primitives are then composed into higher-level constructions like Authenticated Encryption with Additional Data (AEAD) and the NaCl API. All the code in HACL* is verified for memory safety, side channel resistance, and where applicable, also for functional correctness and absence of integer overflow. HACL* code is used as the basis for cryptographic proofs for security in the miTLS project.

HACL* code can be compiled to OCaml using the standard F* compiler, or to C, using the Kremlin backend of F*. The generated C code is as fast as state-of-the-art cryptographic libraries written in C. HACL* is being actively developed on Github; see https://github.com/mitls/hacl-star

6.6. ProScript

Participants: Nadim Kobeissi [correspondant], Karthikeyan Bhargavan, Bruno Blanchet.

Defensive JavaScript (DJS) is a subset of the JavaScript language that guarantees the behaviour of trusted scripts when loaded in an untrusted web page. Code in this subset runs independently of the rest of the JavaScript environment. When propertly wrapped, DJS code can run safely on untrusted pages and keep secrets such as decryption keys. ProScript is a typed subset of JavaScript, inspired by DJS, that is focused on writing verifiable cryptographic protocol implementations. In addition to DJS typing, ProScript imposes a functional style that results in more readable and easily verifiable ProVerif models. ProScript has been used to write and verify a full implementation of the Signal and TLS 1.3 protocols in JavaScript.

The ProScript compiler and various libraries written in ProScript are being developed on Github and will be publicly released in 2017.

6.7. QuickChick

Participants: Maxime Dénès [Inria Sophia-Antipolis], Catalin Hritcu, John Hughes [Chalmers University], Leonidas Lampropoulos [University of Pennsylvania], Zoe Paraskevopoulou [Princeton University], Benjamin Pierce [University of Pennsylvania].

QuickChick is a verified plugin that integrates property-based testing and proving in the Coq proof assistant. This integration is aimed at reducing the cost of formal verification and at providing stronger, formal foundations to property-based testing. https://github.com/QuickChick/QuickChick

6.8. Luck

Participants: Leonidas Lampropoulos [University of Pennsylvania], Diane Gallois-Wong [ENS and Inria Paris], Catalin Hritcu, John Hughes [Chalmers University], Benjamin Pierce [University of Pennsylvania], Li-Yao Xia [ENS and Inria Paris].

Property-based random testing a la QuickCheck requires building efficient generators for well-distributed random data satisfying complex logical predicates, but writing these generators can be difficult and error prone. We propose a domain-specific language in which generators are conveniently expressed by decorating predicates with lightweight annotations to control both the distribution of generated values and the amount of constraint solving that happens before each variable is instantiated. This language, called Luck, makes generators easier to write, read, and maintain. https://github.com/QuickChick/Luck

6.9. Privacy-preserving federated identity

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

security protocols, secure messaging, decentralization, blockchain

Working with the partners in the NEXTLEAP project, we helped design a protocol for decentralized and privacy-preserving identity and key management, creating both a fix privacy vulnerabilities in OAuth (UnlimitID) and on generic versions of blockchain for usages such as key management (ClaimChain). While this software has been prototyped in 2016 using Python working with NEXTLEAP project partner University College London (George Danezis), we expected either formal analysis using ProVerif or verified implementations using Fstar in 2017, as well and integrate this work with formal verification work done in Prosecco on end-to-end secure messaging.

7. New Results

7.1. Verification of Security Protocols in the Symbolic Model

Participants: Bruno Blanchet, Marc Sylvestre.

security protocols, symbolic model, automatic verification 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 [27] 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 as a research report [21] and is submitted to a journal.

Stéphanie Delaune, Mark Ryan, and Ben Smyth [39] introduced the idea of swapping data in order to prove observational equivalence. For instance, ballot secrecy in electronic voting is formalized by saying that A voting a and B voting b is observationally equivalent to (indistinguishable from) A voting b and B voting a. Proving such an equivalence typically requires swapping the votes. However, Delaune et al's approach was never proved correct. Bruno Blanchet and Ben Smyth filled this gap by formalizing the approach and providing a detailed soundness proof [12], [23]. This extension is implemented in ProVerif. Moreover, Marc Sylvestre implemented a graphical display of attacks in ProVerif. The extended tool is available at http://proverif.inria.fr.

Bruno Blanchet wrote a survey on ProVerif, available both as a book and as a journal paper [3].

7.2. Verification of Security Protocols in the Computational model

Participant: Bruno Blanchet.

security protocols, computational model, verification Bruno Blanchet implemented extensions of his computational protocol verifier CryptoVerif. In particular, the tool collects more precise information at each program point, in order to improve the simplification of cryptographic games and the proof of correspondence assertions (authentication). For instance, this extension allows one to prove injective correspondences for protocols with a replay cache. Another extension provides a query to show that several variables are independent secrets. The extended tool is available at http://cryptoverif.inria.fr.

7.3. Verification of Avionic Security Protocols

Participant: Bruno Blanchet.

security protocols, symbolic model, computational model, verification Within the ANR project AnaStaSec, Bruno Blanchet studied an air-ground avionic security protocol, the ARINC823 public key protocol [24]. He 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), he found several weaknesses and imprecisions in the standard. He proposed fixes for these problems. He delivered this work to the ANR and he plans to submit it for publication next year.

7.4. The F* programming language

Participants: Alejandro Aguirre, Danel Ahman [University of Edinburgh], Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud [Microsoft Research], Cédric Fournet [Microsoft Research], Catalin Hritcu, Chantal Keller [Université Paris-Sud], Kenji Maillard, Guido Martínez, Gordon Plotkin, Samin Ishtiaq [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research], Peng Wang [MIT], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoué.

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 Fω (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.

We published a paper on the design, implementation, and formal core of F* at POPL 2016 [20]. A first significant improvement on this design will appear at POPL 2017 under the name of "Dijkstra Monads for Free" [6]. Also significant work was put into extracting a subset of F* to C; we submitted a paper on this to PLDI 2017. F* is being developed as an open-source project at GitHub: https://github.com/FStarLang and the official webpage is at http://fstar-lang.org. We released several beta versions of the software this year.

7.5. Dependable Property-Based Testing

Participants: Maxime Dénès [Inria Sophia-Antipolis], Diane Gallois-Wong [ENS and Inria Paris], Catalin Hritcu, John Hughes [Chalmers University], Leonidas Lampropoulos [University of Pennsylvania], Zoe Paraskevopoulou [Princeton University], Benjamin Pierce [University of Pennsylvania], Li-Yao Xia [ENS and Inria Paris].

This year we finally released the Luck programming language for property-based generators (https://github.com/QuickChick/Luck); a paper on this is about to appear at POPL 2017 [18]. We also improved a previous case study on testing information-flow control mechanisms and published a journal paper on this at JCS [1]. Finally, we kept improving the QuickChick testing plugin for Coq (https://github.com/QuickChick/QuickChick), in particular by automatically producing generators from algebraic datatype definitions.

7.6. Micro-Policies and Secure Compilation

Participants: Arthur Azevedo de Amorim [University of Pennsylvania], André Dehon [University of Pennsylvania and Draper Labs], Catalin Hritcu, Yannis Juglaret, Boris Eng, Benjamin Pierce [University of Pennsylvania], Howard Shrobe [MIT], Stelios Sidiroglou-Douskos [MIT], Greg Sullivan [Draper Labs], Andrew Tolmach [Portland State University].

This year we obtained a new ERC Starting Grant on secure compilation using micro-policies; the grant will start in January 2017. Our work was focused on laying the foundations for this long-term research direction. Preliminary work on this appeared at CSF 2016 [17]. In addition, an improved version of our paper on micro-policies for information flow-control appeared at JFP [1]. Finally, we were part of Draper Labs' patent application on "Techniques for Metadata Processing", as developed jointly in the micro-policies project.

7.7. miTLS: Proofs for TLS 1.3

Participants: Karthikeyan Bhargavan, Chris Brzuska [Technical University of Hamburg], Cedric Fournet [Microsoft Research], Matthew Green [Johns Hopkins University], Markulf Kohlweiss [Microsoft Research], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoué.

transport layer security, cryptographic protocol, verified implementation, man-in-the-middle attack, impersonation attack

We actively participated in the design of TLS 1.3, and worked on a verified implementation of TLS 1.0-1.3 in F*, called miTLS. miTLS is being actively developed on GitHub and we have submitted a paper on our verified implementation of the TLS 1.3 record layer. We published a paper on our overall verification methodology in the IEEE Security and Privacy journal.

Many recent attacks on TLS, discovered by us and others, have relied on *downgrading* a TLS connection and forcing it to use obsolete cryptographic constructions, even if the client and server support and prefer to use modern cryptography. We wrote a paper that showed that such downgrade weaknesses also exist in other protocols such as IPsec, SSH, and ZRTP. We formalized a notion of *downgrade resilience* and showed how it can be achieved in different circumstances. In particular we proved that a new downgrade protection mechanism in TLS 1.3, which was proposed by us, prevents a large class of downgrade attacks. This paper appeared in IEEE S&P (Oakland) 2016 [7].

7.8. Attacks on obsolete cryptography

Participants: Karthikeyan Bhargavan, Gaëtan Leurent.

transport layer security, cryptographic protocol, man-in-the-middle attack, impersonation attack

At NDSS 2016, we published a paper [10] describing a new class of attacks on the use of weak hash functions in popular key exchange protocols such as TLS, IKE, and SSH. One of these attacks, called SLOTH, demonstrated a practical attack on MD5-based client authentication in TLS. We responsibly disclosed this vulnerability, which resulted in security updates in various web browsers and servers. For example, SLOTH-related updates were released for Firefox, Java, RedHat Linux, and for all websites hosted by the Akamai content delivery network.

At CCS 2016, we published a paper [9] that described an attack, called Sweet32, that affects protocols that use block ciphers with short 64-bit blocks, such as Triple-DES and Blowfish. When more than a certain amount of data is sent using such ciphers, the attacker can exploit ciphertext collisions to reconstruct the secret plaintext. We showed how this vulnerability affects TLS and OpenVPN connections. Our findings led to security advisories for OpenVPN, OpenSSL, and all Apple products.

7.9. HACL*: Verified cryptographic library

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

HACL* is a verified cryptographic library written in F*. It implements modern primitives, including elliptic curves like Curve25519, symmetric ciphers like Chacha20, and MAC algorithms like Poly1305. These primitives are then composed into higher-level constructions like Authenticated Encryption with Additional Data (AEAD) and the NaCl API. All the code in HACL* is verified for memory safety, side channel resistance, and where applicable, also for functional correctness and absence of integer overflow. HACL* code is used as the basis for cryptographic proofs for security in the miTLS project.

In CSF 2016, we published a paper on a library of elliptic curves written in F* and compiled to OCaml. This library included the first verified implementations for multiple curves: Curve25519, Curve448, and NIST P-256. However, our code was not very fast. More recently, we worked on Kremlin, a compiler from F* to C that generates code which is as fast as state-of-the-art cryptographic libraries written in C. We have submitted a paper on the Kremlin compiler and its use in HACL*. All our code is being actively and openly developed on GitHub.

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

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

We began work on designing substantial modifications to existing protocols, verifying pre-standard protocols, or creating entirely new standards for new areas. In these areas the fundamental protocols are often unstandardized and controlled by a few large companies (such as the case of identity-based authorization in terms of Google and Facebook's use of OAuth) and new protocols (such as the incompatible space of protocols around secure messaging given by applications such as WhatsApp, Signal, Telegram, and Viber). In some cases, these protocols do not support basic features needed for standardization, such as decentralization and federation. Therefore, in the first half of 2017, Harry Halpin worked with colleagues at IMDEA and University College London in completing the first systematization of knowledge of decentralization, submitted to PETS 2017, and presented preliminary results in "The Responsibility of Open Standards" paper at the HotPETS 2016 workshop as well as in the First Monday journal.

One of the most important protocols in the entire Web is the OAuth protocol, yet it has suffered from a number of dangerous security and privacy issues. Previously formally analysed by Prosecco, one of the larger problems facing this widely deployed protocol is the lack of privacy. Whenever a user log-ins into a website via Google or Facebook Connect (their identity provider), and then authorizes the flow of data between that website and the identity provider. However, the identity provider then gains knowledge of the every single visit that their users make to other websites that request their data, in addition to the data that the identity provider stores itself. Using a new blind signature scheme based on Algebraic MACs, the new UnlimitID protocol makes the use of federated identity by a user at a website unlinkable to their identity provider, while still allowing websites to gain the advantage of single authenticated sign-on to a large identity to prevent spamming and abuse. This work was presented at the Workshop for Privacy in the Electronic Society at ACM CCS. Unlike previous work that requires substantial changes to both websites using OAuth and identity providers, by using the new W3C Web Crypto API (as analyzed by Halpin), this new protocol requires only changes to the identity provider and is do backwards-compatible with existing OAuth implementations. Microsoft has supported this work for possible future standardization in the OpenID Foundation.

In order to be decentralized, secure messaging requires an ability to discover key material and guarantee its integrity. Typically, today this is done via a single centralized and unstandardised service provider. In order to create an interoperable standard around secure messaging, key discovery needs to be decentralized. Blockchain-based approaches have been suggested in previous work in the security research community such as CONIKS, but have failed to take off due to the high deployment cost on centralized servers. We've designed a new protocol, ClaimChain, that builds on both existing work on blockchains while adding new optimizations and providing a decentralized logic based on Rivest and Lampson's SDSI to identify and discovery key material without a trusted third party. Joint work with CNRS to understand the social and economic considerations led to a publication in Internet Science and the existing design will be submitted to a top-notch security conference. Currently, we are discussing early use of this design with codebases used by secure messaging and email providers, and a security and privacy analysis of these codbases was published in CANS. Over the next year we plan for all of these protocols to have formally verified code for their cryptographic functionality and to present a design on how to integrate this work on key discovery into secure messaging with improved privacy and transcript consistency.

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)

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 se´curisation 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)

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.

8.1.1.4. QuickChick

Title: QuickChick: Property-based Testing for Coq

Coordinator: Catalin Hritcu

Abstract: The goal of the project was to develop a property-based testing framework for Coq proofs. Catalin Hritcu was awarded an ANR Jeune Chercheur/Jeune Chercheuse grant to pursue this project, but he declined it in favour of his ERC Starting Grant SECOMP (described below.)

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

Programm: 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 cann 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 Partners

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), Johns Hopkins University, University of Michigan, University of Pennsylvania
- **Web Security**: Microsoft Research (Cambridge, Redmond), Imperial College (London)
- **Micro-Policies**: University of Pennsylvania, Portland State University

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Carmela Troncoso from IMDEA visited the group from 17-18th October and gave a seminar “Traffic Analysis - When Encryption is not Enough to Protect Privacy”

8.4.1.1. Internships

- Alejandro Aguirre: Apr 2016 until Aug 2016
- Abhishek Bichhawat: Sep 2016 until Dec 2016
- Diane Gallois-Wong: Mar 2016 until Aug 2016
- Ritobroto Maitra: May 2016 until Aug 2016
- Guido Martinez: Jan 2016 until Jun 2016
- Jianyang Pan: May 2016 until Aug 2016
- Marina Polubelova: Sep 2016 until Nov 2016
- Natalia Kulatova: May 2016 until Aug 2016
- Vinay Yogendra: May 2016 until Jul 2016

8.4.2. Visits to International Teams

- Bruno Blanchet, March 14 to June 10, 2016, Google, Mountain View.
- Catalin Hritcu, October to November 2016, Microsoft Research, Redmond, USA.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

- Prosecco is organizing the 2nd IEEE European Symposium on Security and Privacy in Paris, | 26-28 April 2017. Catalin Hritcu is General Chair, Bruno Blanchet is Finance Chair, and Karthikeyan Bhargavan is Local arrangements Chair.
- Catalin Hritcu organized two Secure Compilation Meetings (SCM) at Inria Paris in (13 invited participants, 17–19 August 2016) and POPL (15 January 2017)

9.1.2. Scientific Events Selection

9.1.2.1. Member of the Conference Program Committees

- Catalin Hritcu is PC member at POPL 2017
- Catalin Hritcu is PC member at POST 2017
- Catalin Hritcu was PC member at CSF 2016
- Catalin Hritcu was PC member at POST 2016
- Catalin Hritcu was PC member at ITP 2016
- Catalin Hritcu was PC member at cPP 2016
- Harry Halpin is PC member for ACM WWW 2017
- Harry Halpin was PC member for W3C Blockchains and the Web
- Karthikeyan Bhargavan is ERC member at POPL 2017
- Karthikeyan Bhargavan is PC member at IEEE S&P 2017
- Karthikeyan Bhargavan was PC member at IEEE S&P 2016

- Karthikeyan Bhargavan was PC member at ACM CCS 2016
- Karthikeyan Bhargavan was PC member at IEEE CSF 2016
- Karthikeyan Bhargavan was PC member at ACM PLAS 2016

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

- Karthikeyan Bhargavan gave an invited talk at EUROCRYPT 2016
- Karthikeyan Bhargavan gave an invited talk at the OAuth Workshop 2016
- Karthikeyan Bhargavan gave an invited talk at SSTIC 2016

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Master: Bruno Blanchet, Formal Methods, 9h equivalent TD, master M2 MIC, université Paris VII, France
- Master: Bruno Blanchet, Cryptographic protocols: formal and computational proofs, 31.5h equivalent TD, master M2 MPRI, université Paris VII, France
- Master: Karthikeyan Bhargavan, Cryptographic protocols: formal and computational proofs, 31.5h equivalent TD, master M2 MPRI, université Paris VII, France
- Master: Karthikeyan Bhargavan, Protocol Safety and Security, master ACN Telecom ParisTech et Ecole Polytechnique
- Undergraduate: Karthikeyan Bhargavan, INF421 and INF431: Programmation, Ecole Polytechnique
- Doctorat: Karthikeyan Bhargavan: Protecting TLS from legacy cryptography, l'École de printemps en codage et cryptographie, May 2016
- Master: Catalin Hritcu, Cryptographic protocols: formal and computational proofs, 31.5h equivalent TD, master M2 MPRI, université Paris VII, France
- Doctorat: Catalin Hritcu: F* course at Computer Aided Analysis of Cryptographic Protocols summer school, Bucharest, September 2016

9.2.2. Supervision

- PhD completed: Antoine Delignat-Lavaud
  On the Security of Authentication Protocols for the Web
  defended March 2016, supervised by Karthikeyan Bhargavan
- PhD in progress: Evmorfia-Iro Bartzia
  Machine-checked program verification for concrete cryptography,
  defence on February 15, 2017, supervised by Karthikeyan Bhargavan and Pierre-Yves Strub
- PhD in progress: Jean Karim Zinzindohoué
  Analyzing cryptographic protocols and their implementations,
  started September 2014, supervised by Karthikeyan Bhargavan
- PhD in progress: Nadim Kobeissi
  Analyzing cryptographic web applications,
  started February 2015, supervised by Karthikeyan Bhargavan

- PhD incomplete: Yannis Juglaret
  Micro-policies and Secure Compilation,
  started September 2015, interrupted September 2016, supervised by Catalin Hritcu

9.2.3. Juries

- Karthikeyan Bhargavan served on the PhD jury of Olivier Levillain
- Harry Halpin served on the PhD jury of Nikita Mazurov

9.3. Popularization

9.3.1. Seminars

- Karthikeyan Bhargavan: invited talks at SSTIC, EUROCRYPT, OAuth Workshop
- Bruno Blanchet: invited talks at John Mitchell's 60th birthday workshop, Stanford University, CA (May 2016), Facebook, Menlo Park, CA (May 2016), and at University of Oslo (Dec 2016).
- Catalin Hritcu: invited talks at CEA List, MSR Redmond, Inria Gallium, Secure Compilation Meeting, ERC, Inria Prosecco, MPI-SWS
- Harry Halpin: invited talks at NetFutures 2016 (April 2016), Trust in the Digital World (June 2016), Strategic Research Challenges in Privacy-Enhancing Technologies (July 2016), European Dialogue on Internet Governance (September 2016), Internet Governance Forum Tunis (October 2016), Keynote at International Workshop on Semantic Web, and Cryptodesign (November 2016).

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journals

[1] A. AZEVEDO DE AMORIM, N. COLLINS, A. DEHON, D. DEMANGE, C. HRIŢCU, D. PICHARDIE, B. C. PIERCE, R. POLLACK, A. TOLMACH. *A Verified Information-Flow Architecture*, in "Journal of Computer Security (JCS); Special Issue on Verified Information Flow Security", December 2016, vol. 24, n° 6, pp. 689–734 [*DOI :* 10.3233/JCS-15784], https://hal.archives-ouvertes.fr/hal-01424797

[2] K. BHARGAVAN, C. FOURNET, M. KOHLWEISS. *miTLS: Verifying Protocol Implementations against Real-World Attacks*, in "IEEE Security & Privacy", December 2016, vol. 14, n° 6, pp. 18-25 [*DOI :* 10.1109/MSP.2016.123], https://hal.inria.fr/hal-01425964

[3] 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° 1-2, pp. 1 - 135 [*DOI :* 10.1561/3300000004], https://hal.inria.fr/hal-01423760

[4] H. HALPIN, A. MONNIN. *The Decentralization of Knowledge: How Carnap and Heidegger influenced the Web*, in "First Monday", December 2016, vol. 21, n° 12 [*DOI :* 10.5210/FM.V21I12.7109], https://hal.archives-ouvertes.fr/hal-01397931

[5] C. HRIŢCU, L. LAMPROPOULOS, A. SPECTOR-ZABUSKY, A. A. D. AMORIM, M. DÉNÈS, J. HUGHES, B. C. PIERCE, D. VYTINIOTIS. *Testing Noninterference, Quickly*, in "Journal of Functional Programming (JFP); Special issue for ICFP 2013", April 2016, vol. 26, 62 p. , e4 [*DOI :* 10.1017/S0956796816000058], https://hal.archives-ouvertes.fr/hal-01424796

International Conferences with Proceedings

[6] 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 [*DOI :* 10.1145/3009837.3009878], https://hal.archives-ouvertes.fr/hal-01424794

[7] K. BHARGAVAN, C. BRZUSKA, C. FOURNET, M. GREEN, M. KOHLWEISS, S. ZANELLA-BÉGUELIN. *Downgrade Resilience in Key-Exchange Protocols*, in "IEEE Symposium on Security and Privacy (SP), 2016", San Jose, United States, May 2016 [*DOI :* 10.1109/SP.2016.37], https://hal.inria.fr/hal-01425962

[8] K. BHARGAVAN, A. DELIGNAT-LAVAUD, C. FOURNET, A. GOLLAMUDI, G. GONTHIER, N. KOBEISSI, N. KULATOVA, A. RASTOGI, T. SIBUT-PINOTE, N. SWAMY, S. ZANELLA-BÉGUELIN. *Formal Verification of Smart Contracts: Short Paper*, in "ACM Workshop on Programming Languages and Analysis for Security", Vienna, Austria, October 2016 [*DOI :* 10.1145/2993600.2993611], https://hal.inria.fr/hal-01400469

[9] K. BHARGAVAN, G. LEURENT. *On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN*, in "ACM CCS 2016 - 23rd ACM Conference on Computer and Communications Security", Vienna, Austria, ACM, October 2016 [*DOI :* 10.1145/2976749.2978423], https://hal.inria.fr/hal-01404208

[10] K. BHARGAVAN, G. LEURENT. *Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH*, in "Network and Distributed System Security Symposium – NDSS 2016", San Diego, United States, February 2016 [*DOI :* 10.14722/NDSS.2016.23418], https://hal.inria.fr/hal-01244855

[11] K. BHARGAVAN, J. K. ZINZINDOHOUE, E.-I. BARTZIA. *A Verified Extensible Library of Elliptic Curves*, in "29th IEEE Computer Security Foundations Symposium (CSF)", Lisboa, Portugal, June 2016 [*DOI :* 10.1109/CSF.2016.28], https://hal.inria.fr/hal-01425957

[12] B. BLANCHET, B. SMYTH. *Automated Reasoning for Equivalences in the Applied Pi Calculus with Barriers*, in "29th IEEE Computer Security Foundations Symposium (CSF'16)", Lisboa, Portugal, June 2016, pp. 310 - 324 [*DOI :* 10.1109/CSF.2016.29], https://hal.inria.fr/hal-01423742

[13] 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

[14] K. ERMOSHINA, F. MUSIANI, H. HALPIN. *End-to-End Encrypted Messaging Protocols: An Overview*, in "Third International Conference, INSCI 2016 - Internet Science", Florence, Italy, F. BAGNOLI, A. SATSIOU, I. STAVRAKAKIS, P. NESI, G. PACINI, Y. WELP, T. TIROPANIS, D. DIFRANZO (editors), Lecture Notes in Computer Science (LNCS), Springer, September 2016, vol. 9934, pp. 244 - 254 [*DOI :* 10.1007/978-3-319-45982-0_22], https://hal.inria.fr/hal-01426845

[15] H. HALPIN. *The Responsibility of Open Standards in the Era of Surveillance*, in "Hot Topics in Privacy Enchancing Technologies", Darmstadt, Germany, HotPETS, July 2016, https://hal.inria.fr/hal-01426848

[16] 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",

Vienna, Austria, WPES, October 2016, pp. 139 - 142 [*DOI :* 10.1145/2994620.2994637], https://hal.inria.fr/hal-01426847

[17] Y. JUGLARET, C. HRIŢCU, A. A. D. 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)", Lisabon, Portugal, IEEE Computer Society Press, 2016, pp. 45–60 [*DOI :* 10.1109/CSF.2016.11], https://hal.archives-ouvertes.fr/hal-01424795

[18] 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 [*DOI :* 10.1145/3009837.3009868], https://hal.archives-ouvertes.fr/hal-01424793

[19] E. SPARROW, H. HALPIN, K. KANEKO, R. POLLAN. *LEAP: A next-generation client VPN and encrypted email provider*, in "Proceedings of Cryptology and Network Security (CANS)", Milan, Italy, Lecture Notes in Computer Science (LNCS), Springer, November 2016, vol. 10052., pp. 176 - 191 [*DOI :* 10.1007/978-3-319-48965-0_11], https://hal.inria.fr/hal-01426850

[20] 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)", St. Petersburg, Florida, United States, ACM, 2016, pp. 256-270 [*DOI :* 10.1145/2837614.2837655], https://hal.archives-ouvertes.fr/hal-01265793

Research Reports

[21] M. ABADI, B. BLANCHET, C. FOURNET. *The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication*, ArXiv, September 2016, 110 p. , https://hal.inria.fr/hal-01423924

[22] K. BHARGAVAN, A. DELIGNAT-LAVAUD, N. KOBEISSI. *A Formal Model for ACME: Analyzing Domain Validation over Insecure Channels*, Inria Paris ; Microsoft Research Cambridge, November 2016, https://hal.inria.fr/hal-01397439

[23] B. BLANCHET, B. SMYTH. *Automated reasoning for equivalences in the applied pi calculus with barriers*, Inria Paris, April 2016, n° RR-8906, 54 p. , https://hal.inria.fr/hal-01306440

References in notes

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

[25] 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

[26] 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

[27] 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

[28] 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.

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

[30] 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.

[31] 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

[32] 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

[33] 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° 1, pp. 3–51

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

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

[36] 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

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

[38] 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

[39] S. DELAUNE, M. D. RYAN, B. SMYTH. *Automatic verification of privacy properties in the applied pi-calculus*, in "IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security", International Federation for Information Processing (IFIP), Springer, 2008, vol. 263, pp. 263–278

[40] 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

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

[42] 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

[43] 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

[44] N. SWAMY, J. CHEN, C. FOURNET, P.-Y. STRUB, K. BHARGAVAN, J. YANG. *Secure distributed programming with value-dependent types*, in "J. Funct. Program.", 2013, vol. 23, n^{o} 4, pp. 402-451

[45] 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