RESEARCH CENTRE

Inria Centre
at Université Côte d'Azur

2023
ACTIVITY REPORT

Project-Team
STAMP

Safety Techniques based on Formalized Mathematical Proofs

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Inria

Contents

Project-Team STAMP

Creation of the Project-Team: 2019 November 01

Keywords

Computer sciences and digital sciences

A2.1.11. – Proof languages

A2.4.3. – Proofs

A4.5. – Formal methods for security

A7.2. – Logic in Computer Science

A7.2.3. – Interactive Theorem Proving

A7.2.4. – Mechanized Formalization of Mathematics

A8.3. – Geometry, Topology

A8.4. – Computer Algebra

A8.10. – Computer arithmetic

Other research topics and application domains

B6.1. – Software industry

B9.5.1. – Computer science

B9.5.2. – Mathematics

1 Team members, visitors, external collaborators

Research Scientists

- Yves Bertot [Team leader, INRIA, Senior Researcher, HDR]
- Cyril Cohen [INRIA, Researcher]
- Benjamin Grégoire [INRIA, Researcher, until Jun 2023]
- Laurence Rideau [INRIA, Researcher]
- Enrico Tassi [INRIA, Researcher]
- Laurent Théry [INRIA, Researcher]

Post-Doctoral Fellows

- Pierre Boutry [INRIA, until Aug 2023]
- Jean-Christophe Léchenet [INRIA, until Sep 2023]
- Paolo Torrini [INRIA, Post-Doctoral Fellow, from Jun 2023]

PhD Students

- Davide Fissore [UNIV COTE D'AZUR, from Oct 2023]
- Swarn Priya [UNIV COTE D'AZUR, until Jan 2023]
- Quentin Vermande [UNIV COTE D'AZUR, from Oct 2023]

Technical Staff

- Maxime Dénès [INRIA, Engineer, until Aug 2023]
- Thomas Portet [INRIA, Engineer, from Feb 2023]
- Romain Tetley [INRIA, Engineer, from Oct 2023]

Administrative Assistant

- Christine Foggia [INRIA, from Mar 2023]

2 Overall objectives

Computers and programs running on these computers are powerful tools for many domains of human activities. In some of these domains, program errors can have enormous consequences. It will become crucial for all stakeholders that the best techniques are used when designing these programs.

We advocate using higher-order logic proof assistants as tools to obtain better quality programs and designs. These tools make it possible to build designs where all decisive arguments are explicit, ambiguity is alleviated, and logical steps can be verified precisely. In practice, we are intensive users of the Coq system and we participate actively to the development of this tool, in collaboration with other teams at Inria, and we also take an active part in promoting its usage by academic and industrial users around the world.

Many domains of modern computer science and engineering make a heavy use of mathematics. If we wish to use proof assistants to avoid errors in designs, we need to develop corpora of formally verified mathematics that are adapted to these domains. Developing libraries of formally verified mathematics is the main motivation for our research. In these libraries, we wish to capture not only the knowledge

that is usually recorded in definitions and theorems, but also the practical knowledge that is recorded in mathematical practice, idioms, and work habits. Thus, we are interested in logical facts, algorithms, and notation habits. Also, the very process of developing an ambitious library is a matter of organization, with design decisions that need to be evaluated and improved. Refactoring of libraries is also an important topic. Among all higher-order logic based proof assistants, we contend that those based on Type theory are the best suited for this work on libraries, thanks to their strong capabilities for abstraction and modular re-use.

The interface between mathematics, computer science and engineering is large. To focus our activities, we will concentrate on applications of proof assistants to two main domains: cryptography and robotics. We also develop specific tools for proofs in cryptography, mainly around a proof tool named EasyCrypt.

3 Research program

3.1 Theoretical background

The proof assistants that we consider provide both a programming language, where users can describe algorithms performing tasks in their domain of interest, and a logical language to reason about the programs, thus making it possible to ensure that the algorithms do solve the problems for which they were designed. Trustability is gained because algorithms and logical statements provide multiple views of the same topic, thus making it possible to detect errors coming from a mismatch between expected and established properties. The verification process is itself a logical process, where the computer can bring rigor in aligning expectations and guarantees.

The foundations of proof assistants rest on the very foundations of mathematics. As a consequence, all aspects of reasoning must be made completely explicit in the process of formally verifying an algorithm. All aspects of the formal verification of an algorithm are expressed in a discourse whose consistency is verified by the computer, so that unclear or intuitive arguments need to be replaced by precise logical inferences.

One of the foundational features on which we rely extensively is *Type Theory*. In this approach a very simple programming language is equiped with a powerful discipline to check the consistency of usage: types represent sets of data with similar behavior, functions represent algorithms mapping types to other types, and the consistency can be verified by a simple computer program, a *type-checker*. Although they can be verified by a simple program, types can express arbitrary complex objects or properties, so that the verification work lives in an interesting realm, where verifying proofs is decidable, but finding the proofs is undecidable.

This process for producing new algorithms and theorems is a novelty in the development of mathematical knowledge or algorithms, and new working methods must be devised for it to become a productive approach to high quality software development. Questions that arise are numerous. How do we avoid requiring human assistance to work on mundane aspects of proofs? How do we take advantage of all the progress made in automatic theorem proving? How do we organize the maintenance of ambitious corpora of formally verified knowledge in the long term?

To acquire hands-on expertise, we concentrate our activity on three aspects. The first one is foundational: we develop and maintain a library of mathematical facts that covers many aspects of algebra and analysis. In the past, we applied this library to proofs in group theory, but it is increasingly used for many different areas of mathematics and by other teams around the world, from combinatorics to elliptic cryptography, for instance. The second aspect is applicative: we develop a specific tool for proofs in cryptography, where we need to reason on the probability that opponents manage to access information we wish to protect. For this activity, we develop a specific proof system, relying on a wider set of automatic tools, with the objective of finding the tools that are well adapted to this domain and to attract users that are initially specialists in cryptography but not in formal verification. The third domain is robotics, as we believe that the current trend towards more and more autonomous robots and vehicles will raise questions of safety and trustability where formal verification can bring significant added value.

4 Application domains

4.1 Mathematical Components

The Mathematical Components library is the main by-product of an effort started almost two decades ago to provide a formally verified proof for a major theorem in group theory. Because this major theorem had a proof published in books of several hundreds of pages, with elements coming from character theory, other coming from algebra, and some coming from real analysis, it was an exercise in building a large library, with results in many domains, and in establishing clear guidelines for further increase and data search.

This library has proved to be a useful repository of mathematical facts for a wide area of applications, so that it has a growing community of users in many countries (Denmark, France, Germany, Japan, Singapore, Spain, Sweden, UK, USA) and for a wide variety of topics (transcendental number theory, elliptic curve cryptography, articulated robot kinematics, recently block chain foundations).

Interesting questions on this library range around the importance of decidability and proof irrelevance, the way to structure knowledge to automatically inherit theorems from one topic to another, the way to generate infrastructure to make this automation efficient and predictable. In particular, we want to concentrate on adding a new mathematical topic to this library: real analysis and then complex analysis (Mathematical Components Analysis).

On the front of automation, we are convinced that a higher level language is required to describe similarities between theories, to generate theorems that are immediate consequences of structures, etc, and for this reason, we invest in the development of a new language on top of the proof assistant (ELPI, Embeddable Lambda Prolog Interpreter).

4.2 Proofs in cryptography

When we work on cryptography, we are interested in the formal verification of proofs showing that some cryptographic primitives provide good guarantees against unwanted access to information. Over the years we have developed a technique for this kind of reasoning that relies on a programing logic (close to Hoare logic) with probabilistic aspects and the capability to establish relations between several implementations of a problem. The resulting programming logic is called *probabilistic relational Hoare logic*. We also study questions of *side-channel* attacks, where we wish to guarantee that opponents cannot gain access to protected knowledge, even if they observe specific features of execution, like execution time (to which the answer lies in *constant-time* execution) or partial access to memory bits (to which the answer lies in *masking*).

For this domain of application, we choose to work with a specific proof tool (EasyCrypt), which combines powerful first-order reasoning and use of automatic tools, with a specific support for probabilistic relational Hoare Logic. The development of this EasyCrypt proof tool is one of the objectives of our team.

When it comes to formal proofs of resistance to side-channel attacks, we contend that it is necessary to verify formally that the compiler used in the production of actually running code respects the resistance properties that were established in formally verified proofs. One of our objectives is to develop such a compiler (Jasmin) and show its strength on a variety of applications.

The pair of tools EasyCrypt and Jasmin has also proved its worth in the formal verification of correctness for post-quantum cryptography.

4.3 Proofs for robotics

Robots are man-made artifacts where numerous design decisions can be argued based on logical or mathematical principles. For this reason, we wish to use this domain of application as a focus for our investigations. The questions for which we are close to providing answers involve precision issues in numeric computation, obstacle avoidance and motion planning (including questions of graph theory), articulated limb kinematics and dynamics, and balance and active control.

From the mathematical perspective, these topics require that we improve our library to cover real algebraic geometry, computational geometry, real analysis, graph theory, and refinement relations between abstract algorithms and executable programs.

In the long run, we hope to exhibit robots where pieces of software and part of the design have been subject to formal verification.

5 Highlights of the year

5.1 Awards

- Swarn Priya has been one of the winners of the Young Talents "Pour les femmes et la science" L'Oréal-UNESCO 2023 prize. She defended her PhD [18] in November.
- The paper ¨Typing High-Speed Cryptography against Spectre v1" [10] has obtained a Distinguished Paper Award at the IEEE Symposium on Security and Privacy.

6 New software, platforms, open data

6.1 New software

6.1.1 Coq

Name: The Coq Proof Assistant

Keywords: Proof, Certification, Formalisation

Scientific Description: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an integrated development environment (IDE).

Functional Description: Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

Release Contributions: An overview of the new features and changes, along with the full list of contributors is available at https://coq.inria.fr/refman/changes.html#version-8-18 .

News of the Year: Coq version 8.18 integrates changes to several parts of the system : kernel, specification language, type inference, notation, tactics, Ltac2 language, commands and options, command-line tools, CoqIDE, standard library, infrastructure and dependencies, extraction. See https://coq.inria.fr/refman/changes.html#version-8-18 for an overview of the new features and changes, along with the full list of contributors.

URL: http://coq.inria.fr/

Contact: Matthieu Sozeau

Participants: Yves Bertot, Frédéric Besson, Tej Chajed, Cyril Cohen, Pierre Corbineau, Pierre Courtieu, Maxime Dénès, Jim Fehrle, Julien Forest, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Georges Gonthier, Benjamin Grégoire, Jason Gross, Hugo Herbelin, Vincent Laporte, Olivier Laurent, Assia Mahboubi, Kenji Maillard, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pedrot, Clément Pit-Claudel, Kazuhiko Sakaguchi, Vincent Semeria, Michael Soegtrop, Arnaud Spiwack, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Anton Trunov, Li-Yao Xia, Theo Zimmermann

Partners: CNRS, Université Paris-Sud, ENS Lyon, Université Paris-Diderot

6.1.2 coq-elpi

Keywords: Metaprogramming, Extension

Scientific Description: Coq-elpi provides a Coq plugin that embeds ELPI. It also provides a way to embed Coq terms into lambdaProlog using the Higher-Order Abstract Syntax approach (HOAS) and a way to read terms back. In addition to that it exports to ELPI a set of Coq primitives, e.g. printing a message, accessing the environment of theorems and data types, defining a new constant and so on. For convenience it also provides a quotation and anti-quotation for Coq's syntax in lambdaProlog. E.g. {{nat}} is expanded to the type name of natural numbers, or {{A -> B}} to the representation of a product by unfolding the -> notation. Finally it provides a way to define new vernacular commands and new tactics.

Functional Description: Coq plugin embedding ELPI

Release Contributions: - parsing/execution separation

News of the Year: - Separation of parsing/execution for modern UIs. - Application of type-checking to solve type class instances using an ELPI program. - Application of coercion to insert explicit type casts using an ELPI program.

Publications: hal-01897468, hal-01637063

Contact: Enrico Tassi

Participants: Enrico Tassi, Davide Fissore

6.1.3 ELPI

Name: Embeddable Lambda Prolog Interpreter

Keywords: Constraint Programming, Programming language, Higher-order logic

Scientific Description: The programming language has the following features

- Native support for variable binding and substitution, via a Higher Order Abstract Syntax (HOAS) embedding of the object language. The programmer does not need to care about technical devices to handle bound variables, like De Bruijn indices.

- Native support for hypothetical context. When moving under a binder one can attach to the bound variable extra information that is collected when the variable gets out of scope. For example when writing a type-checker the programmer needs not to care about managing the typing context.

- Native support for higher-order unification variables, again via HOAS. Unification variables of the meta-language (lambdaProlog) can be reused to represent the unification variables of the object language. The programmer does not need to care about the unification-variable assignment map and cannot assign to a unification variable a term containing variables out of scope, or build a circular assignment.

- Native support for syntactic constraints and their meta-level handling rules. The generative semantics of Prolog can be disabled by turning a goal into a syntactic constraint (suspended goal). A syntactic constraint is resumed as soon as relevant variables get assigned. Syntactic constraints can be manipulated by constraint handling rules (CHR).

- Native support for backtracking, to ease implementation of search.

- The constraint store is extensible. The host application can declare non-syntactic constraints and uses custom constraint solvers to check their consistency.

- Clauses are graftable. The user is free to extend an existing program by inserting/removing clauses, both at runtime (using implication) and at "compilation" time by accumulating files.

Most of these features come with lambdaProlog. Constraints and propagation rules are novel in ELPI.

Functional Description: ELPI implements a variant of lambdaProlog enriched with Constraint Handling Rules, a programming language well suited to manipulate syntax trees with binders and unification variables.

ELPI is a research project aimed at providing a programming platform for the so called elaborator component of an interactive theorem prover.

ELPI is designed to be embedded into larger applications written in OCaml as an extension language. It comes with an API to drive the interpreter and with an FFI for defining built-in predicates and data types, as well as quotations and similar goodies that come in handy to adapt the language to the host application.

Release Contributions: - Faster separate compilation/linking

News of the Year: - Time complexity improvement of separate compilation/linking of program units, which is now pseudo linear. - The runtime was made re-entrant, allowing multiple Elpi instances to live in the same process. - New deep-indexing data structure based on discrimination trees.

URL: https://github.com/lpcic/elpi/

Publications: hal-03800154, hal-01176856, hal-01410567, hal-01897468

Contact: Enrico Tassi

Participants: Enrico Tassi, Claudio Sacerdoti Coen

6.1.4 Easycrypt

Keywords: Proof assistant, Cryptography

Functional Description: EasyCrypt is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs. EasyCrypt can also be used for reasoning about differential privacy.

Release Contributions: This version introduces a new logic (ehoare) allowing to bound the expectation of a function in a probabilistic program.

News of the Year: The major release (2023.09) has been published. This release include the a new logic for bounding the expectation of function in a probabilistic program.

URL: https://github.com/EasyCrypt/easycrypt

Publications: hal-03352062, hal-03469015

Contact: Gilles Barthe

Participants: Benjamin Grégoire, Gilles Barthe, Pierre-Yves Strub, Adrien Koutsos

6.1.5 Hierarchy Builder

Keywords: Coq, Metaprogramming

Scientific Description: It is nowadays customary to organize libraries of machine checked proofs around hierarchies of algebraic structures. One influential example is the Mathematical Components library on top of which the long and intricate proof of the Odd Order Theorem could be fully formalized. Still, building algebraic hierarchies in a proof assistant such as Coq requires a lot of manual labor and often a deep expertise in the internals of the prover. Moreover, according to our experience, making a hierarchy evolve without causing breakage in client code is equally tricky: even a simple refactoring such as splitting a structure into two simpler ones is hard to get right. Hierarchy Builder is a high level language to build hierarchies of algebraic structures and to make these hierarchies evolve without breaking user code. The key concepts are the ones of

factory, builder and abbreviation that let the hierarchy developer describe an actual interface for their library. Behind that interface the developer can provide appropriate code to ensure retro compatibility. We implement the Hierarchy Builder language in the hierarchy-builder addon for the Coq system using the Elpi extension language.

Functional Description: Hierarchy Builder is a high level language for Coq to build hierarchies of algebraic structures and to make these hierarchies evolve without breaking user code. The key concepts are the ones of factory, builder and abbreviation that let the hierarchy developer describe an actual interface for their library. Behind that interface the developer can provide appropriate code to ensure retro compatibility.

Release Contributions: Support for hierarchy of morphisms and bugfixes. Adding compatibility with Coq 8.16

News of the Year: - Major performance improvements in handling large hierarchies (e.g. MathComp 2.0)

URL: https://github.com/math-comp/hierarchy-builder

Publication: hal-02478907

Contact: Enrico Tassi

Participants: Enrico Tassi, Cyril Cohen

Partners: University of Tsukuba, Onera

6.1.6 Jasmin

Name: Jasmin compiler and analyser

Keywords: Cryptography, Static analysis, Compilers

Functional Description: The Jasmin programming language smoothly combines high-level and low-level constructs, so as to support “assembly in the head” programming. Programmers can control many low-level details that are performance-critical: instruction selection and scheduling, what registers to spill and when, etc. The language also features high-level abstractions (variables, functions, arrays, loops, etc.) to structure the source code and make it more amenable to formal verification. The Jasmin compiler produces predictable assembly and ensures that the use of high-level abstractions incurs no run-time penalty.

The semantics is formally defined to allow rigorous reasoning about program behaviors. The compiler is formally verified for correctness (the proof is machine-checked by the Coq proof assistant). This ensures that many properties can be proved on a source program and still apply to the corresponding assembly program: safety, termination, functional correctness...

Jasmin programs can be automatically checked for safety and termination (using a trusted static analyzer). The Jasmin workbench leverages the EasyCrypt toolset for formal verification. Jasmin programs can be extracted to corresponding EasyCrypt programs to prove functional correctness, cryptographic security, or security against side-channel attacks (constant-time).

Release Contributions: 2023.06.0 is a major release of Jasmin. It contains a few noteworthy changes: - local functions now use call and ret instructions, - experimental support for the ARMv7 (i.e., Cortex-M4) architecture, - a few aspects of the safety checker can be finely controlled through annotations or command-line flags, - shift and rotation operators have a simpler semantics.

As usual, it also brings in various fixes and improvements, such as bit rotation operators and automatic slicing of the input program.

News of the Year: On June 2023, a major release (2023.06.0) has been published.

URL: https://github.com/jasmin-lang/jasmin

Publications: hal-04106448, hal-04218417, hal-03844366, hal-03430789, hal-03352062, hal-02404581, hal-02974993, hal-01649140

Contact: Jean-Christophe Léchenet

Participants: Gilles Barthe, Benjamin Grégoire, Adrien Koutsos, Vincent Laporte, Jean-Christophe Léchenet, Swarn Priya, Santiago Arranz Olmos

Partners: The IMDEA Software Institute, Ecole Polytechnique, Universidade do Minho, Universidade do Porto, Max Planck Institute for Security and Privacy

6.1.7 Trocq

Keywords: Proof synthesis, Proof transfer, Coq, Elpi, Logic programming, Parametricity, Univalence

Functional Description: Trocq is a prototype of a modular parametricity plugin for Coq, aiming to perform proof transfer by translating the goal into an associated goal featuring the target data structures as well as a rich parametricity witness from which a function justifying the goal substitution can be extracted.

The plugin features a hierarchy of parametricity witness types, ranging from structure-less relations to a new formulation of type equivalence, gathering several pre-existing parametricity translations, including univalent parametricity and CoqEAL, in the same framework.

This modular translation performs a fine-grained analysis and generates witnesses that are rich enough to preprocess the goal yet are not always a full-blown type equivalence, allowing to perform proof transfer with the power of univalent parametricity, but trying not to pull in the univalence axiom in cases where it is not required.

The translation is implemented in Coq-Elpi and features transparent and readable code with respect to a sequent-style theoretical presentation.

News of the Year: We released the first version of Trocq, for demo purposes and to support the claims made in the associated paper. Trocq is able to translate non trivial goals between isomorphic or partially isomorphic representations.

URL: https://github.com/coq-community/trocq

Publication: hal-04177913

Contact: Cyril Cohen

Participants: Cyril Cohen, Enzo Crance, Assia Mahboubi

Partner: Mitsubishi Electric R&D Centre Europe, France

6.1.8 VsCoq

Name: VsCoq

Keywords: Coq, User Interfaces

Functional Description: VsCoq is an extension for Visual Studio Code (VS Code) and VSCodium which provides support for the Coq Proof Assistant.

VsCoq is distributed in two flavours:

- VsCoq Legacy (required for Coq < 8.18, compatible with Coq >= 8.7) is based on the original VsCoq implementation by C.J. Bell. It uses the legacy XML protocol spoken by CoqIDE.

- VsCoq (recommended for Coq >= 8.18) is a full reimplementation around a language server which natively speaks the LSP protocol.

Release Contributions: We have mainly been working on stability and bug fixes, in this release you'll find :

- Some improvements to performance on large files. - Fixing document state invalidation bugs. - Goal view improvements.

News of the Year: The first version (2.0.1) of VsCoq based on the LSP protocol has been released on September, 2023.

URL: https://github.com/coq-community/vscoq

Contact: Laurent Théry

7 New results

7.1 Formal proof of post-quantum cryptographic primitive

Participants: José Bacelar Almeida *(INESC TEC)*, Manuel Barbosa *(University of Porto & INESC TEC)*, Gilles Barthe *(MPI-SP & IMDEA)*, Christian Doczkal *(MPI-SP)*, Jelle Don *(Centrum Wiskunde & Informatica)*, François Dupressoir *(University of Bristol)*, Serge Fehr *(Leiden University)*, Benjamin Grégoire, Yu-Hsuan Huang *(Leiden University)*, Andreas Hülsing *(Eindhoven University)*, Vincent Laporte *(Pesto)*, Yi Lee *(MPI-SP)*, Jean-Christophe Léchenet, Matthias Meijers *(Eindhoven University)*, Tiago Oliveira *(MPI-SP)*, Hugo PachecoUniversidade do Minho & INESC TEC] , Miguel Quaresma *(MPI-SP)*, Peter Schwabe *(MPI-SP & Radboud University)*, Antoine Séré *(LIX)*, Pierre-Yves Strub *(PQShield)*, Xiaodi Wu *(University of Maryland)*.

In July 2022, NIST announced the first batch of "winners" of the post-quantum project, i.e., schemes that will be forwarded to standardization [25]. This first batch contained three signature schemes (CRYSTALS-Dilithium [28, 32], Falcon [33], and SPHINCS$^+$ [26, 29]), and only one key-encapsulation mechanism (KEM): the lattice-based scheme CRYSTALS-Kyber [27, 30].

We have started the formal verification of three of those primitives (CRYSTALS-Dilithium, SPHINCS$^+$, and CRYSTALS-Kyber) following different directions.

For Kyber, we give a (readable) formal specification in the EasyCrypt proof assistant, which is syntactically very close to the pseudocode description of the scheme as given in the most recent version of the NIST submission. We also provide high-assurance open-source implementations of Kyber written in the Jasmin language, along with machine-checked proofs that they are functionally correct with respect to the EasyCrypt specification. To make this possible it was necessary to extend the Jasmin language. This work has been published in [11].

For CRYSTALS-Dilithium and SPHINCS$^+$, instead of proving the functional correctness of an implementation we have started to prove the semantic security of the schemes, i.e the correctness of the specification. The work on SPHINCS$^+$ has been published in [17]. The work on CRYSTALS-Dilithium has been published in [12]. To accomplish this work we have extended EasyCrypt with a new logic allowing to bound the expectation of a function in a probabilistic program.

7.2 CryptoVerif to EasyCrypt

Participants: Bruno Blanchet *(Prosecco)*, Pierre Boutry, Christian Doczkal *(MPI-SP)*, Benjamin Gre'goire, Pierre-Yves Strub *(PQShield)*.

We continue our study of approaches to combine two mechanized tools to verify protocols. We developed a translation from CryptoVerif to EasyCrypt that allows cryptographic assumptions that cannot

be proved in CryptoVerif to be translated to EasyCrypt and proved there. We used the translation to prove different hypotheses assumed in CryptoVerif:

- The reduction of the N query formulation of the Computational/Gap Diffie-Hellman (CDH/GDH) games in CryptoVerif to the standard, single-query formulation. The obtained bounds are better than what can be obtained by a direct hybrid argument.

- The reduction from the N participant games (e.g. insider or outsider adversaries) for authenticated Key encapsulation mechanisms (KEM) to 1 or 2 participant games.

We completed the translation to cover a wider range of the language that CryptoVerif uses for specifying assumptions on cryptographic primitives. This work [22] has been accepted for publication in CSF 2024.

7.3 Extending the Jasmin compiler

Participants: Basavesh Ammanaghatta Shivakumar *(MPI-SP)*, Santiago Arranz Olmos *(MPI-SP)*, Gilles Barthe *(MPI-SP & IMDEA)*, Benjamin Grégoire, Vincent Laporte *(Pesto)*, Jean-Christophe Léchenet, Tiago Oliviera *(MPI-SP)*, Swarn Priya, Peter Schwabe *(MPI-SP & Radboud University)*, Lucas Tabary-Maujean *(ENS Paris-Saclay)*.

We have extended Jasmin with a new back-end for arm-v7. The main difficulty was to generalize the compiler to be independent from the architecture (different pointer size, different calling convention, different instruction set and so on). Before that, the only back-end was for x86-64 with avx2 extension. This generalization is an important step, because it will allow to easily add other back-ends, in particular we plan to add RISC-V.

The language has been extended with new features for security.

- The compiler can introduce code that zeroizes the stack at the end of export functions. Three strategies are currently supported: ‘unrolled‘ (the code is a sequence of writes as long as needed), ‘loop‘ (the code is a loop) and ‘loopSCT‘ (same as ‘loop‘ but with a ‘LFENCE‘ at the end to defend against Spectre attacks).

- Protection against Spectre attacks: We have proposed, analyzed, implemented and evaluated an approach for writing efficient cryptographic implementations that are protected against Spectre v1 attacks in Jasmin. Our approach ensures speculative constant-time. Speculative constant-time is enforced by means of a (value-dependent) information flow type system and the use of primitives allowing to implement speculative load hardening protection. This work has been published in [10].

7.4 Collisions between trajectories and polygonal obstacles

Participants: Yves Bertot, Laurent Théry.

In an effort to synthesize several years of investigations around the computation of robot trajectories, we developed a Coq model for a program that takes as input a description of obstacles and a pair of points and produces as output a trajectory for a robot from one point to the other between these obstacles. The obstacles are given by a collection of straight line segments and the produced trajectory is composed of straight line segments and Bézier curves, so that the trajectory is smooth. An article describing the different phases of the program is submitted for publication [21]. This Coq mod l is actually a program that can be run inside Coq. Thanks to the extraction tool, the same program can also be run in a web page. Proofs of correctness for this program are under construction.

7.5 Instance saturation in Hierarchy Builder

Participants: Yves Bertot, Cyril Cohen, Thomas Portet, Enrico Tassi.

In the initial revision of Hierarchy Builder, definitions needed to be added in a precise order, otherwise instances of structures would be missing in the final graph of inheritance. We developed an extension that verifies all the instances that would be missing and includes them. Thanks to this extension the Hierarchy Builder program is more robust, as the user does not need to respect a specific order of definitions anymore.

7.6 New type class solver

Participants: Davide Fissore, Enrico Tassi.

We are developing a new type class solver for Coq by compiling type class instances into rules for the Elpi programming language. Currently we are validating a prototype implementation on the Std++ and TLC Coq libraries, two widely used libraries that rely on type classes.

7.7 Automation for separation logic in Coq

Participants: Davide Fissore, Enrico Tassi, Robbert Krebbers *(Radboud University)*, Ike Mulder *(Radboud University)*.

We are trying to use the Elpi programming language to automate proofs in separation logic. Diaframe is an existing automatic prover based on Coq type classes that suffers from the limitations of the current Coq solver. We have improved the indexing data structures used by Elpi in order to make them scale to larger inputs. Also, we are trying to use partial evaluation in order to specialize, ahead of time, the rules used for type class search in the context of separation logic.

7.8 Porting to Mathcomp 2

Participants: Reynald Affeldt *(AIST Japan)*, Yves Bertot, Cyril Cohen, Pierre Roux *(Onera)*, Kazuhiko Sakaguchi *(Galinette)*, Enrico Tassi.

We ported the entire Mathcomp ecosystem to the new major release (version 2) of the mathematical components library. Most software have been released, and Mathcomp analysis and Abel are ported but not released yet. The details of the port are described in [19]

7.9 Trocq : Proof Transfer for Free

Participants: Cyril Cohen, Enzo Crance *(Galinette)*, Assia Mahboubi *(Galinette)*
.

In interactive theorem proving, a range of different representations may be available for a single mathematical concept, and some proofs may rely on several representations. Without automated support such as proof transfer, theorems available with different representations cannot be combined, without manual input from the user. Tools with such a purpose exist, but in proof assistants based on dependent

type theory, it still requires human effort to prove transfer, whereas it is obvious and often left implicit on paper. We present Trocq, a new proof transfer framework, based on a generalization of the univalent parametricity translation, thanks to a new formulation of type equivalence. This translation takes care to avoid dependency on the axiom of univalence for transfers in a delimited class of statements, and may be used with relations that are not necessarily isomorphisms. We motivate and apply our framework on a set of examples designed to show that it unifies several existing proof transfer tools. The article [23] also discusses an implementation of this translation for the Coq proof assistant, in the Coq-Elpi metalanguage.

7.10 VsCoq: a user interface for Coq

Participants: Maxime Dénès, Thomas Portet, Enrico Tassi, Romain Tetley, Laurent Théry.

A rewrite of the VSCoq extension has been completed this year. This leads to the publication of the release V2.0.1 in September. This effort is meant to continue for a few years and provide a modern and stable user interface for Coq. Maxime Dénès and Enrico Tassi have worked in regular sprints since February, helping Romain Tetley to dive into the Coq language server.

7.11 Handling subsets and subtypes in Hierarchy Builder

Participants: Cyril Cohen, Quentin Vermande.

We are experimenting with new design patterns to automate the conversion between sets and types, to automatically prove set membership and to automatically cast between types even when an external proof is required. The result of these experiments will be integrated in Hierarchy Builder in order to extend its expressiveness, in particular in the formalization of topology, number theory and category theory.

This is ongoing work without any publication yet. Early experiments were presented during meetings of the Liberabaci projects.

7.12 Handling enriched categories in Hierarchy Builder

Participants: Cyril Cohen, Enrico Tassi, Paolo Torrini.

We have been working since June 2023 on the CoREACT project, which addresses the development of applied category theory in Coq. Our workload involves using the Hierarchy Builder (HB) and improving it to match the project goal. HB is useful in the formalization of complex algebraic hierarchies, making it possible to automate inheritance and to manage efficiently hierarchy evolution relative to a type subject. First we extended HB in order to support reasoning about enriched categories. In fact, the subject localization associated with enrichment has made it necessary to implement an appropriate connector that we call wrapper, allowing the user to benefit from the automation provided by HB without having to resort to mathematically unnatural formulations. Part of our initial work also involved clarifying the operational meaning of wrapping with respect to the informal semantics of HB. Then, since October, we moved on to formalize categorical theories that make use of related notions, notably double categories and internal categories. We have currently provided two alternative characterizations of double categories and we are proving their equivalence, as part of the development of a Coq library.

7.13 Abel Galois Theorem

Participants: Cyril Cohen, Quentin Vermande.

We extended the Abel-Galois theorem to the case of the positive characteristic. This involved the generalization of several definitions and lemmas, and in particular the contribution of Hilbert Theorem 90 in its additive version.

7.14 Lebesgue measure and integration

Participants: Reynald Affeldt *(AIST Japan)*, Cyril Cohen.

The construction of the Lebesgue integral and its measure has been completed and published in [9]. This paper describes the techniques of formalization that were needed to obtained comfortably usable definitions.

7.15 Building finite fields via irreducible polynomials

Participants: Cyril Cohen, Joshua Cohen *(Princeton University)*, Laurent Théry.

We have introduced a construction for finite fields in the Mathcomp Library. We have first defined polynomials of a given size from which we have derived the standard module structure. Then, we use the theory of irreducible polynomials to get to finite fields. This contribution has been added to Mathcomp version 1.19.

7.16 Formal study of double-word arithmetic algorithms

Participants: Tom Hubrecht *(ENS Paris)*, Claude-Pierre Jeannerot *(Aric)*, Vincent Lefèvre *(Aric)*, Nicolas Louvet *(ENS Lyon)*, Jean-Michel Muller *(CNRS)*, Joris Picot *(ENS Lyon)*, Laurence Rideau, Laurent Théry, Paul Zimmermann *(Caramba)*.

We have continued our collaboration inside the ANR `Nuscap` about double-word arithmetic. First, an article on the work on the formalization of algorithms for euclidian norm has been published [8]. Second, we have started a formalization in Coq+Flocq of the proofs given in [34] (with an extended version in [35]). This paper describes algorithms for the correct rounding of the power function x^y in the binary64 `IEEE 754` format, for all rounding modes. We have verified (and amended with the help of the authors) all the paper proofs given in the article. The formal proofs are available on github. For this work we also had to formalize the correctness of the FastTwoSum algorithm with directed roundings given in [36].

7.17 Formal study of the Fast Fourier Transform

Participants: Nicolas Brisebarre *(CNRS)*, Laurence Rideau, Laurent Théry.

We have continued our collaboration inside the ANR `Nuscap` about the Fast Fourier Algorithm. First we have a formal proof of the relative error of the Cooley-Tukey Fast Fourier Transform given in [31]. Second, we have developed a certified Fast Fourier algorithm that is executable inside Coq. It uses a complex-number interval arithmetic built on top of the Coq interval library. It is used to get a toy implementation of a multiplication algorithm for complex-number polynomials.

7.18 Simplification of a constructive version of Tarski's system of geometry

Participants: Pierre Boutry.

In work that was started in Pierre Boutry's thesis, we study how Tarski's work on axioms for reasoning in geometry can be made constructive. This is a follow-up of work on the same topic from 2020. We progressed on the independence of the new axioms. The current state has been presented at ADG [13].

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

Participants: Benjamin Grégoire, Swarn Priya, Yves Bertot.

The STAMP team participates with the Grace team (Inria Saclay) in the JASMIN contract funded in the framework of the Inria-Nomadic Labs collaboration for research related to the Tezos blockchain. This contract funds the PhD thesis of Swarn Priya.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Inria associate team not involved in an IIL or an international program

FLAVOR

Participants: Yves Bertot, Cyril Cohen, Laurence Rideau, Enrico Tassi, Laurent Théry.

Title: Formal Library of Analysis for the Verification of Robots

Duration: 2020 ->

Coordinator: Reynald Affeldt (reynald.affeldt@aist.go.jp)

Partners:

- National Institute of Advanced Industrial Science and Technology Tokyo (Japan)

Inria contact: Yves Bertot

Summary: The objective is to apply formal methods based on Coq to software and designs that are concerned with robots. Covered topics concern mathematical formalization for real analysis, control theory, kinematic chains, and motion planning.

9.2 International research visitors

9.2.1 Visits of international scientists

International visits to the team

Robbert Krebbers

Status Professor

Institution of origin: Radboud University Nijmegen

Country: the Netherlands

Dates: June 12-16, 2023

Context of the visit: Work on merging mathematical components and Stdpp libraries

Mobility program/type of mobility: research stay and lecture

9.3 National initiatives

9.3.1 ANR

- Scrypt "Compilation sécurisée de primitives cryptographiques" started on February 1st, 2019, for 48 months, with a grant of 100 kEuros. Other partners are Inria team Celtique (Inria Rennes Bretagne Atlantique), Ecole polytechnique, and AMOSSYS SAS. The corresponding researcher for this contract is Benjamin Grégoire. This action was used to fund post-doctoral researchers.

- NuSCAP "Numerical Safety for Computer-Aided Proofs", started on February 1st, 2021 for 48 months, with a grant covering travel costs. Other partners are CNRS-LIP, Sorbonne University LIP6, and CNRS-LAAS. The corresponding researcher for this contract is Laurence Rideau.

- CoREACT “Coq-based Rewriting: towards Executable Applied Category Theory”, started on March 1st, 2023, for 48 months, with a grant of 67,3 kEuros for STAMP, funding a post-doc, instruments and material costs and travel costs. Other partners are IRIF (Université Paris Cité), LIP (ENS-Lyon) and LIX (École Polytechnique). The corresponding researcher for this contract is Cyril Cohen.

9.3.2 PEPR

- SVP PEPR Cybersecurity. We participate in a project concerned with the verification of security protocols. Partners in this project are CNRS IRISA Rennes (coordinator Stéphanie Delaune), Inria, University of Paris-Saclay, University of Lorraine, University of Côte d’Azur, ENS Rennes. The funds allocated to our team in this collaboration are 333 kEuros. The corresponding researcher for this contract is Benjamin Grégoire. This action will be used to fund researchers (doctoral students or post-doctoral researchers).

9.3.3 Inria Challenges

- Liber Abaci. Yves Bertot coordinates the Inria challenge Liber Abaci on the use of a Type-theory based proof assistant to improve mathematics education for the first years of higher education (undergraduate mathematics).

10 Dissemination

Participants: Yves Bertot, Pierre Boutry, Cyril Cohen, Benjamin Grégoire, Enrico Tassi, Laurence Rideau.

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

General chair, scientific chair

- Yves Bertot and Enrico Tassi organized and chaired the program committee for the Coq workshop in July in Bialystok.

- Cyril Cohen is in the steering committee of ITP (Interactive Theorem Proving) since September.
- Cyril Cohen has been workshop chair for the conferences CPP 2023 (Certified Programs and Proofs), CADE 29 (Conference on Automated Deduction) and ITP 2923 (Interactive Theorem Proving).
- Enrico Tassi is in the steering committee of LFMTP (Logical Frameworks and Metalanguages, Theory and Practice).
- Enrico Tassi has been member of COQ 2023 (Coq Workshop), COQPL 2024 (Coq for Programming Languages), CADE 29 (Conference on Automated Deduction) and CPP 2024 (Certified Programs and Proofs).

Reviewer

- Pierre Boutry did a review for ThEdu 2024 (Theorem proving and Education).
- Yves Bertot reviewed a chapter for a book in honor of Herman Geuvers.

10.1.2 Invited talks

- Benjamin Grégoire gave an invited talk at the GDR-SI (Sécurité Informatique) in June.
- Yves Bertot gave an invited talk at the conference ThEdu, Theorem Proving in Education in Rome in July.

10.1.3 Research administration

Cyril Cohen has created and is now a co-administrator of the Coq Zulip chat.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Yves Bertot and Nicolas Magaud (Université de Strasbourg) gave a course on using Coq to teach mathematics at the summer school "Proof assistants and teaching" in Val d'Ajol in June.
- Yves Bertot gave a course on using Coq for mathematics at the summer school "Interactions of Proof Assistants and Mathematics" in Regensburg, Germany, in September. The teaching material for these courses is available here.
- Yves Bertot gave an introductory course on Coq in the framework of Inria Academy in October.
- Pierre Boutry and Julien Narboux (Université de Strasbourg) gave a tutorial on GeoCoq [14] at ADG 2023.

10.2.2 Supervision

- Enrico Tassi has co-supervised (with Jesper Bentson) the master thesis "Expanding Coq with Type Aware Code Completion" by Hjalte Dalland, Jakob Israelsen and Simon Kristensen, ITU Copenhagen.
- Enrico Tassi has supervised the master thesis "Type-class solver in Type Theory via Logic Programming" by Davide Fissore, UCA-DS4H, until Davide Fissore registered for a PhD program.
- Yves Bertot and Cyril Cohen supervise the thesis of Quentin Vermande (Université Côte d'Azur) starting in September.
- Yves Bertot and Enrico Tassi supervise the thesis of Davide Fissore (Université Côte d'Azur) starting in October.
- Yves Bertot and Benjamin Grégoire supervised the thesis of Swarn Priya (Université Côte d'Azur) until November.

10.2.3 Juries

- Yves Bertot was member of the jury with report duty (rapporteur) for Rebecca Zucchini (University of Paris-Saclay) in June.
- Yves Bertot was member of the jury for Mohit Tekriwal (University of Michigan at Ann Arbor) in June and for Loïc Germerie-Guizouarn (Université Côte d'Azur) in December.
- Enrico Tassi was member of the jury for Enzo Crance (Université de Nantes) in December.

10.3 Popularization

10.3.1 Internal or external Inria responsibilities

Laurence Rideau is member of the editorial board of Interstices.

10.3.2 Articles and contents

- Yves Bertot wrote an article for a technical magazine aimed at programming hobbyists and professionals. This article is available as a preprint on hal [24].

11 Scientific production

11.1 Major publications

[1] B. Ammanaghatta Shivakumar, G. Barthe, B. Grégoire, V. Laporte, T. Oliveira, S. Priya, P. Schwabe and L. Tabary-Maujean. 'Typing High-Speed Cryptography against Spectre v1'. In: *2023 IEEE Symposium on Security and Privacy (SP)*. SP 2023- IEEE Symposium on Security and Privacy. San Francisco, United States, May 2023, pp. 1592–1609. DOI: 10.1109/SP46215.2023.10179418. URL: https://hal.science/hal-04106448.

[2] S. Bernard, C. Cohen, A. Mahboubi and P.-Y. Strub. 'Unsolvability of the Quintic Formalized in Dependent Type Theory'. In: ITP 2021 - 12th International Conference on Interactive Theorem Proving. Rome / Virtual, France, 29th June 2021. URL: https://inria.hal.science/hal-03136002.

[3] C. Cohen, K. Sakaguchi and E. Tassi. 'Hierarchy Builder: algebraic hierarchies made easy in Coq with Elpi'. In: FSCD 2020 - 5th International Conference on Formal Structures for Computation and Deduction. 167. Paris, France, 2020, 34:1–34:21. DOI: 10.4230/LIPIcs.FSCD.2020.34. URL: https://inria.hal.science/hal-02478907.

[4] B. Grégoire, J.-C. Léchenet and E. Tassi. 'Practical and sound equality tests, automaticallyDeriving eqType instances for Jasmin's data types with Coq-Elpi'. In: CPP '23: 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. CPP 2023: Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. Boston MA USA, France: ACM, 16th Jan. 2023, pp. 167–181. DOI: 10.1145/3573105.3575683. URL: https://inria.hal.science/hal-03800154.

[5] J.-M. Muller and L. Rideau. 'Formalization of double-word arithmetic, and comments on "Tight and rigorous error bounds for basic building blocks of double-word arithmetic"'. In: *ACM Transactions on Mathematical Software* 48.1 (Mar. 2022), pp. 1–24. DOI: 10.1145/3484514. URL: https://hal.science/hal-02972245.

11.2 Publications of the year

International journals

[6] R. Affeldt and C. Cohen. 'Measure Construction by Extension in Dependent Type Theory with Application to Integration'. In: *Journal of Automated Reasoning* 67.3 (Sept. 2023), p. 28. DOI: 10.1007/s10817-023-09671-5. URL: https://inria.hal.science/hal-04183173.

[7] M. Barbosa, G. Barthe, B. Grégoire, A. Koutsos and P.-Y. Strub. 'Mechanized Proofs of Adversarial Complexity and Application to Universal Composability: Journal pre-print: full version'. In: *ACM Transactions on Privacy and Security* 26.3 (30th Aug. 2023), pp. 1–34. DOI: 10.1145/3589962. URL: https://inria.hal.science/hal-04048217.

[8] V. Lefèvre, N. Louvet, J.-M. Muller, J. Picot and L. Rideau. 'Accurate calculation of Euclidean Norms using Double-word arithmetic'. In: *ACM Transactions on Mathematical Software* 49.1 (21st Mar. 2023), pp. 1–34. DOI: 10.1145/3568672. URL: https://hal.science/hal-03482567.

International peer-reviewed conferences

[9] R. Affeldt, C. Cohen and A. Saito. 'Semantics of Probabilistic Programs using s-Finite Kernels in Coq'. In: CPP 2023 - Certified Programs and Proofs. CPP 2023: Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. Boston, United States, 2023. DOI: 10.1145/3573105.3575691. URL: https://inria.hal.science/hal-03917948.

[10] B. Ammanaghatta Shivakumar, G. Barthe, B. Grégoire, V. Laporte, T. Oliveira, S. Priya, P. Schwabe and L. Tabary-Maujean. 'Typing High-Speed Cryptography against Spectre v1'. In: *2023 IEEE Symposium on Security and Privacy (SP)*. SP 2023- IEEE Symposium on Security and Privacy. San Francisco, United States, May 2023, pp. 1592–1609. DOI: 10.1109/SP46215.2023.10179418. URL: https://hal.science/hal-04106448.

[11] J. Bacelar Almeida, M. Barbosa, G. Barthe, B. Grégoire, V. Laporte, J.-C. Léchenet, T. Oliveira, H. Pacheco, M. Quaresma, P. Schwabe, A. Séré and P.-Y. Strub. 'Formally verifying Kyber: Episode IV: Implementation correctness'. In: *ACR Transactions on Cryptographic Hardware and Embedded Systems*. CHES 2023 - Conference on Cryptographic Hardware and Embedded Systems. Vol. 2023. 3. Praha, Czech Republic, 9th June 2023, pp. 164–193. DOI: 10.46586/tches.v2023.i3.164-193. URL: https://inria.hal.science/hal-04218417.

[12] M. Barbosa, G. Barthe, C. Doczkal, J. Don, S. Fehr, B. Grégoire, Y.-H. Huang, A. Hülsing, Y. Lee and X. Wu. 'Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium'. In: *Lecture Notes in Computer Science*. CRYPTO 2023 - 43rd International Cryptology Conference. Vol. LNCS-14085. Advances in Cryptology – CRYPTO 2023 : 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part V. Santa Barbara, United States, 20th Aug. 2023, pp. 358–389. DOI: 10.1007/978-3-031-38554-4_12. URL: https://hal.science/hal-04315311.

[13] P. Boutry, S. Kastenbaum and C. Saintier. 'Towards an Independent Version of Tarski's System of Geometry'. In: 14th International Conference on Automated Deduction in Geometry. Belgrade, Serbia, 20th Sept. 2023. URL: https://hal.science/hal-04324071.

[14] P. Boutry and J. Narboux. 'Tutorial Laboratory - GeoCoq to formalize high-school geometry problems'. In: ADG 2023 - Automated Deduction in Geometry 2023. Belgrade, Serbia, 2023. URL: https://hal.science/hal-04230732.

[15] B. Grégoire, J.-C. Léchenet and E. Tassi. 'Practical and sound equality tests, automatically – Deriving eqType instances for Jasmin's data types with Coq-Elpi'. In: CPP '23: 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. CPP 2023: Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. Boston MA USA, France: ACM, 16th Jan. 2023, pp. 167–181. DOI: 10.1145/3573105.3575683. URL: https://inria.hal.science/hal-03800154.

Scientific book chapters

[16] Y. Bertot and L. C. Paulson. 'Inductive Predicates'. In: *Proof Assistants and Their Applications to Mathematics and Computer Science*. 04. 2023, p. 37. URL: https://inria.hal.science/hal-04311869.

Edition (books, proceedings, special issue of a journal)

[17] M. Barbosa, F. Dupressoir, B. Grégoire, A. Hülsing, M. Meijers and P.-Y. Strub, eds. *Machine-Checked Security for XMSS as in RFC 8391 and SPHINCS +*. 2023. DOI: 10.1007/978-3-031-38554-4_14. URL: https://hal.science/hal-04315335.

Doctoral dissertations and habilitation theses

[18] S. Priya. 'Formally computer-verified protections against timing-based side-channel attacks'. Université Côte d'Azur, 22nd Nov. 2023. URL: https://hal.science/tel-04331805.

Reports & preprints

[19] R. Affeldt, Y. Bertot, C. Cohen, P. Roux, K. Sakaguchi and E. Tassi. *Porting Coq Scripts to the Mathematical Components Library Version 2*. Inria Sophia Antipolis - Méditerranée, Université Côte d'Azur; National Institute of Advanced Industrial Science and Technology (AIST), Japan; ONERA / DTIS, Université de Toulouse, France, 20th June 2023, pp. 1–12. URL: https://hal.science/hal-04225130.

[20] X. Allamigeon, Q. Canu, C. Cohen, K. Sakaguchi and P.-Y. Strub. *Design patterns of hierarchies for order structures*. 28th Feb. 2023. URL: https://inria.hal.science/hal-04008820.

[21] Y. Bertot. *Safe smooth paths between straight line obstacles*. 28th Nov. 2023. URL: https://inria.hal.science/hal-04312815.

[22] B. Blanchet, P. Boutry, C. Doczkal, B. Grégoire and P.-Y. Strub. *CV2EC: Getting the Best of Both Worlds*. 4th Dec. 2023. URL: https://inria.hal.science/hal-04321656.

[23] C. Cohen, E. Crance and A. Mahboubi. *Trocq: Proof Transfer for Free, With or Without Univalence*. 12th July 2023. URL: https://hal.science/hal-04177913.

11.3 Other

Scientific popularization

[24] Y. Bertot. 'Prouvez que vos programmes fonctionnels n'ont pas de bugs avec Coq Première partie'. In: *Programmez !* 256 (2023), p. 35. URL: https://inria.hal.science/hal-04219914.

11.4 Cited publications

[25] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone and Y.-K. Liu. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. NISTIR 8413. https://csrc.nist.gov/publications/detail/nistir/8413/final. 2022.

[26] J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe and B. Westerbaan. *SPHINCS$^+$ – Submission to the NIST post-quantum project, v.3.1*. Round-3 submission to the NIST PQC standardization project. https://sphincs.org/data/sphincs+-r3.1-specification.pdf. 2022.

[27] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé. CRYSTALS-KYBER*: Algorithm Specifications And Supporting Documentation (version 3.02)*. Round-3 submission to the NIST PQC standardization project. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf. 2021.

[28] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehlé. CRYSTALS-DILITHIUM*: Algorithm Specifications and Supporting Documentation (Version 3.1)*. Round-3 submission to the NIST PQC standardization project. https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf. 2021.

[29] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld and P. Schwabe. 'The SPHINCS$^+$ Signature Framework'. In: *ACM CCS 2019: 26th*. Ed. by L. Cavallaro, J. Kinder, X. Wang and J. Katz. London, UK, Nov. 2019, pp. 2129–2146. DOI: 10.1145/3319535.3363229.

[30] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe and D. Stehlé. 'CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM'. In: *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*. https://eprint.iacr.org/2017/634. IEEE, 2018, pp. 353–367.

[31] N. Brisebarre, M. Joldes, J.-M. Muller, A.-M. Naneş and J. Picot. 'Error analysis of some operations involved in the Cooley-Tukey Fast Fourier Transform'. In: *ACM Transactions on Mathematical Software* 46.2 (May 2020), pp. 1–34. DOI: 10.1145/3368619. URL: https://hal.science/hal-01949458.

[32] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehlé. 'CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme'. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.1 (2018). https://tches.iacr.org/index.php/TCHES/article/view/839, pp. 238–268. DOI: 10.13154/tches.v2018.i1.238-268.

[33] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte and Z. Zhang. FALCON*: Fast-Fourier Lattice-based Compact Signatures over NTRU (Specification v1.2)*. Round-3 submission to the NIST PQC standardization project. https://falcon-sign.info/falcon.pdf. 2020.

[34] T. Hubrecht, C.-P. Jeannerod and P. Zimmermann. 'Towards a correctly-rounded and fast power function in binary64 arithmetic'. In: *2023 IEEE 30th Symposium on Computer Arithmetic (ARITH 2023)*. Vol. 2023 IEEE 30th Symposium on Computer Arithmetic (ARITH). Portland, Oregon (USA), United States, Sept. 2023. URL: https://inria.hal.science/hal-04326201.

[35] T. Hubrecht, C.-P. Jeannerod and P. Zimmermann. 'Towards a correctly-rounded and fast power function in binary64 arithmetic, extended version'. This is the extended version of an article published in the proceedings of ARITH 2023. July 2023. URL: https://inria.hal.science/hal-04159652.

[36] P. Zimmermann. 'Note on FastTwoSum with Directed Roundings'. working paper or preprint. Sept. 2023. URL: https://inria.hal.science/hal-03798376.