Activity Report 2015

Project-Team PROSECCO

Programming securely with cryptography

RESEARCH CENTER
Paris - Rocquencourt

THEME
Security and Confidentiality

Table of contents

Project-Team PROSECCO

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

Keywords:

Computer Science and Digital Science:

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
2.1.11. - Proof languages
2.1.3. - Functional programming
2.1.7. - Distributed programming
2.2. - Compilation
2.2.1. - Static analysis
2.2.3. - Run-time systems
2.4. - Reliability, certification
2.4.2. - Verification
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, from Nov 2015]
Catalin Hritcu [Inria, Researcher]

Engineers

Gergely Bana [Inria, granted by FP7 PROSECO- ERC CRYSP project]
Benjamin Beurdouche [Inria, granted by FP7 PROSECO- ERC CRYSP project]

PhD Students

Antoine Delignat-Lavaud [Inria, until Sep 2015, granted by FP7 PROSECO- ERC CRYSP project]
Yannis Juglaret [Inria, from Mar 2015]
Nadim Kobeissi [Inria, from Feb 2015]
Jean-Karim Zinzindohoué [Min. Ecologie]

Administrative Assistant

Anna Bednarik [Inria]

Others

Simon Forest [ENS Paris, until Aug 2015]
Li-Yao Xia [ENS Paris, until Aug 2015]
Alfredo Pironti [Politecnico di Torino]
Santiago Zanella Beguelin [MSR-Inria]

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 three key focus areas for our research in the short- to medium term.

2.1.1. Symbolic verification of cryptographic applications

Our goal is 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.

2.1.2. 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 Computational F7 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.

2.1.3. Provably secure web applications

We plan 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 [47]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [43] 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* [38]. 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 [35], [22], correspondences (including authentication) [36], and observational equivalences [34]. 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 [31], JFK [23], and Web Services Security [33], 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 [40], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [41]. 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# [31]. 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 [29] 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 [32]. F* [52] 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 [44] 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. 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 [30] [30] and TS* [54], 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.

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 15 articles in international peer-reviewed journals and conferences, including papers in prestigious conferences such as IEEE S&P Oakland (2 papers), ACM CCS, NDSS, WWW, ASPLOS, and ITP, and we won four research awards for our work, detailed below.

We released updates to F*, miTLS, ProVerif, and CryptoVerif, along with our collaborators at other institutions. We discovered serious vulnerabilities in a number of TLS libraries, web browsers, and web servers, resulting in several published CVEs, and over a dozen software updates based on our recommendations in widely used software such as Firefox, Chrome, Internet Explorer, Safari, OpenSSL, Java, and Mono.

5.1.1. Awards

- Distinguished paper award, IEEE Symposium for Security and Privacy, 2015
- Best student paper award, ACM Conference on Computer and Communications Security, 2015
- Best paper award, Usenix Workshop on Offensive Technologies, 2015
- Pwnie award for Most Innovative Research, BlackHat USA, 2015

6. New Software and Platforms

6.1. ProVerif

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

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–].

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], Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Alfredo Pironti, Pierre-Yves Strub [IMDEA], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoue.

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.

A paper describing the miTLS library was published at IEEE S&P 2013, CRYPTO 2014, and several updates to the software were released in 2015. The software and associated research materials are available from http://mitls.org.

6.4. flexTLS

Participants: Karthikeyan Bhargavan [correspondant], Alfredo Pironti, Benjamin Beurdouche.

flexTLS is a TLS testing framework based on miTLS, and is released as part of the miTLS distribution. Unlike miTLS, flexTLS can be configured to run incorrect TLS clients and servers in order to test other TLS implementations. Using flexTLS we analyzed a series of open source TLS implementations and found important vulnerabilities like SKIP and FREAK. We also used flexTLS to build proof-of-concept demos for other attacks such as Logjam.

A paper describing flexTLS was published at Usenix WOOT 2015. The software and associated research materials are available from http://mitls.org.

6.5. F*

Participants: Nikhil Swamy [Microsoft Research], Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Catalin Hritcu, Chantal Keller, Aseem Rastogi, Pierre-Yves Strub.

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 $F\omega$ (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 [53]. 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.

6.6. 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 TextSecure protocol in JavaScript.

The ProScript compiler and various libraries written in ProScript will be made available from the Prosecco webpage.

7. New Results

7.1. Verification of Security Protocols in the Symbolic Model

Participants: Bruno Blanchet, Miriam Paiola.

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 [24] does not come with proofs, and detailed proofs for the results in this paper were never published. This year, Martín Abadi, Bruno Blanchet, and Cédric Fournet finished the detailed proofs of all results of this paper, started last year, and added a new example on a symbolic analog of indifferentiability of hash functions. This work is submitted to a journal.

Previously [37], Bruno Blanchet and Miriam Paiola presented an automatic technique for proving secrecy and authentication properties for security protocols that manipulate lists of unbounded length, for an unbounded number of sessions. That work relies on an extension of Horn clauses, generalized Horn clauses, designed to support unbounded lists, and on a resolution algorithm on these clauses. However, in that previous work, they had to model protocols manually with generalized Horn clauses, which is unpractical. They recently extended the input language of ProVerif to model protocols with lists of unbounded length. They give the formal meaning of this extension, translate it automatically to generalized Horn clauses, and prove that this translation is sound. This work appears as a research report [21].

We implemented several extensions of ProVerif: Bruno Blanchet and Vincent Cheval improved the algorithm for proving observational equivalence between two processes, by merging them into a single biprocess that encodes the two processes. Bruno Blanchet also introduced a new construct **new** $a[x_1, ..., x_n]$ in ProVerif which allows to specify the arguments $x_1, \cdots, x_n$ used in the internal representation of the fresh name a. This extension allows one to tune the precision and speed of the analysis performed by ProVerif. The extended tool is available at http://proverif.inria.fr, and deposited to the APP (*Agence pour la Protection des Programmes*).

Stéphanie Delaune, Mark Ryan, and Ben Smyth [42] 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. They plan to submit this work to a conference.

7.2. Verification of Security Protocols in the Computational model

Participant: Bruno Blanchet.

Bruno Blanchet implemented several extensions of his computational protocol verifier CryptoVerif. In particular, he improved the global dependency analysis, used in order to show that the result of all tests is independent from some random values. He improved the proof of secrecy properties, in particular to prove forward secrecy properties. He also improved the merging of branches of tests, in particular to be able to merge the two branches of **if** b **then** P_1 **else** P_2 even when variables are renamed between P_1 and P_2. Finally, he added the display of an explanation of why a cryptographic transformation fails, to make the tool easier to use. The extended tool is available at http://cryptoverif.inria.fr.

Within the ANR project AnaStaSec, Bruno Blanchet verified an air-ground avionic security protocol (International Civil Aviation Organization (ICAO) Document 9880: Manual on Detailed Technical Specifications for the Aeronautical Telecommunication Network (ATN) using ISO/OSI standards and protocols, Part IV) using CryptoVerif. He proved entity authentication and message authenticity for the main protocol, in the computational model of cryptography, and made comments on some points that should be clarified in the protocol specification. He presented this work at a meeting of the secure dialog service working group of ICAO, in Toulouse, September 2015. The working group was strongly interested by the presentation and welcomed the proposal to apply these modelling and formal verification techniques as part of its validation activities.

7.3. The F* programming language

Participants: Nikhil Swamy [Microsoft Research], Catalin Hritcu, Chantal Keller [LRI], Aseem Rastogi [Univ of Maryland], Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cedric Fournet [Microsoft Research], Pierre-Yves Strub [IMDEA], Markulf Kohlweiss [Microsoft Research], Jean Karim Zinzindohoue, Santiago Zanella Beguelin [Microsoft Research, MSR-Inria].

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. 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.

7.4. Micro-Policies and Secure Compilation

Participants: Catalin Hritcu, Arthur Azevedo de Amorim, Zoi Paraskevopoulou, Nikolaos Giannarakis.

Following on from previous work on the *micro-policy* framework, Catalin Hritcu and his collaborators published new work on applications and efficient implementations of micro-policies. They published work on low-level implementations of micro-policies at ASPLOS 2015 [18]. At IEEE S&P, they published a paper how to write formally verified reference monitors using micro-policies [26].

Other than these published works, Hritcu and his colleagues also worked on using micro-policies to enforce secure information flow at the hardware level [25], and a secure compiler for a high-level language that relies on micro-policies to enforce programming language abstractions [45].

7.5. Dependable Property-Based Testing

Participants: Catalin Hritcu, Zoi Paraskevopoulou.

Catalin Hritcu and his student, Zoi Paraskevopoulou, worked on a methodology for formally verified property-based testing and implemented it as a foundational verification framework for QuickChick, a port of QuickCheck to Coq. This work was published at ITP 2015 [19]. Catalin Hritcu also worked with a number of co-authors on a new technique for creating random generators for property-based testing. This work is currently under submission [46].

7.6. Attacks and Proofs for Transport Layer Security

Participants: Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Alfredo Pironti, Pierre-Yves Strub [IMDEA], Jean Karim Zinzindohoue.

As a countermeasure to our earlier work on the triple handshake attack, we proposed a TLS extension called *session hash* which has now been published as an Internet standard (IETF RFC 7627). We also formally analyzed various protocols such as TLS, IKE, and SSH for key synchronization and triple handshake attacks, and proved that our session hash countermeasure prevents such attacks on TLS. This work appeared at NDSS 2015 [15].

We discovered and reported an important class of *state machine attacks* on implementations of the Transport Layer Security (TLS) protocol. These attacks appear when TLS implementations incorrectly accept messages which are forbidden by the TLS state machine. We built a test framework for such attacks and analyzed a number of open source implementations. Our analysis uncovered critical vulnerabilities such as the SKIP attack on Java and the FREAK attack on almost all mainstream web browsers. The research results were published at IEEE S&P where our paper won a distinguished paper award [14]. Our work also led to security updates and CVEs for many web browsers, TLS libraries, and web servers.

Along with colleagues at several other institutions, we discovered the Logjam vulnerability on protocols that still support weak Diffie-Hellman groups in their key exchange. We showed that the attack could be used for online and offline attacks on real-world TLS clients and servers. We also showed how the vulnerability could weaken the security of IPsec and SSH connections. Our research led to widespread changes to the configurations of web servers, mail servers, web browsers, and TLS libraries. The research was published at ACM CCS 2015 [12] where it won a Best Paper award.

Antoine Delignat-Lavaud showed how the unsafe sharing of certificates across multiple HTTPS websites could be exploited to fully compromise the same origin policy for websites, using a vulnerability called *virtual host confusion*. A research paper on these attacks appeared at WWW 2015 [17].

7.7. Privacy, Electronic Voting, and Auctions

Participants: Benjamin Smyth [correspondant], Elizabeth Quaglia.

Benjamin Smyth worked on a formal analysis of privacy in Direct Anonymous Attestation schemes [50]. He also showed how to verify commitment protocols in ProVerif without False attacks [39].

Apart from these published works, Benjamin Smyth and Elizabeth Quaglia worked on formal security analyses of electronic auction schemes based on existing models for electronic voting [48]. Benjamin Smyth worked on developing new formal definitions for secrecy and independence in election schemes [51], and on applying such definitions to the security analysis of real-world voting protocols such as Helios and JCJ [49].

7.8. Computationally Complete Symbolic Attacker Models

Participants: Gergei Bana, Hubert Comon-Lundh [ENS Cachan], Rohit Chadha [University of Missouri].

In previous work, Bana and Comon-Lunch proposed a new approach to computational verification of cryptographic protocols, by defining a *computationally complete* symbolic attacker, so that a symbolic proof against this attacker can be shown to imply a computational proof of security [27], [28].

Following on from this work, Bana and Chadha fully developed the core parts of the computationally complete symbolic attacker based on indistinguishability. This covers both trace properties and equivalence properties and can be proved partially complete. They evaluated their method by applying it to several classic protocols. This work is currently under submission.

Bana, Comon-Lundh, and Koutsos also worked on a decision procedure for the computationally complete symbolic attacker based on indistinguishability.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Grants with Industry

The miTLS project received a grant from Mozilla for work on TLS 1.3. Catalin Hritcu received a PhD grant from Microsoft Research.

9. Partnerships and Cooperations

9.1. National Initiatives

9.1.1. ANR

9.1.1.1. ProSe

Title: ProSe: Security protocols : formal model, computational model, and implementations (ANR VERSO 2010.)

Other partners: Inria/Cascade, ENS Cachan-Inria/Secsi, LORIA-Inria/Cassis, Verimag.

Duration: December 2010 - December 2014.

Coordinator: Bruno Blanchet, Inria (France)

Abstract: The goal of the project is to increase the confidence in security protocols, and in order to reach this goal, provide security proofs at three levels: the symbolic level, in which messages are terms; the computational level, in which messages are bitstrings; the implementation level: the program itself.

9.1.1.2. AJACS

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

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

Duration: October 2014 - March 2019.

Coordinator: Alan Schmitt, Inria (France)

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

9.1.2. FUI

9.1.2.1. Pisco

Title: PISCO

Partners: Bull, Cassadian, CEA, CS, Saferiver, Serpikom, Telecom Paristech

Duration: January 2013 - December 2014.

Coordinator: Liliana Calabanti, Bull (France)

Abstract: The goal of the project is to develop a prototype of a new secure applicance based on a virtual machine architecture accessing an HSM. The role of PROSECCO is to contribute to the analysis of security http://www.systematic-paris-region.org/en/projets/pisco

9.2. European Initiatives

9.2.1. FP7 & H2020 Projects

9.2.1.1. CRYSP

Title: CRYSP: A Novel Framework for Collaboratively Building Cryptographically Secure Programs and their Proofs

Programm: FP7

Duration: November 2010 - October 2015

Coordinator: Inria

Inria contact: Anne-Lise Chenet-Pflieger

The goal of CRYSP is to use recent advances in software verification and dependent type systems and apply them to the verification of cryptographic protocol implementations written in a variety of languages. We want to enable the collaborative development of such programs and their specifications. Our target is to be able to verify mainstream implementations of the Transport Layer Security Protocol.

9.3. International Initiatives

9.3.1. Inria International Partners

9.3.1.1. Informal International Partners

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

- **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

9.4. International Research Visitors

9.4.1. Visits of International Scientists

- Deepak Garg from the Max Planck Institute for Software Systems in Saarbruecken visited the group from 10-12 June and gave a seminar.
- Udit Dhawan from the University of Pennsylvania visited the group from 10-14 March and gave a seminar.
- Cedric Fournet and Nikhil Swamy from Microsoft Research visited the group multiple times to work on joint projects.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific events selection

10.1.1.1. Member of the conference program committees

- ACM POPL – January 2015, Mumbai, India: Karthikeyan Bhargavan
- ACM CCS – October 2015, Denver, USA: Karthikeyan Bhargavan
- ETAPS POST – April 2015, London, UK: Karthikeyan Bhargavan
- ACM SEC@SAC – April 2015, Salamanca, Spain: Karthikeyan Bhargavan
- HotSpot – April 2015, London, UK: Bruno Blanchet

10.1.1.2. Reviewer

- Members of Prosecco reviewed papers for many major conferences and workshops including ACM CCS, ACM POPL, IEEE S&P, etc.

10.1.2. Journal

10.1.2.1. Member of the editorial boards

Associate Editor

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

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Master: Bruno Blanchet, Cryptographic protocols: formal and computational proofs, 9h equivalent TD, master M2 MPRI, université Paris VII, France

License: Karthikeyan Bhargavan, INF431, INF421, INF672, INF321, introductory courses at at Ecole Polytechnique, Palaiseau, France

Doctorat: Catalin Hritcu, lectures at University of Saarbruecken, Germany

Doctorat: Catalin Hritcu, F* tutorials at POPL 2015 and ICFP 2015

10.2.2. Supervision

PhD in progress: Evmorfia-Iro Bartzia
Machine-checked program verification for concrete cryptography,
started October 2011, supervised by Karthikeyan Bhargavan and Pierre-Yves Strub

PhD in progress: Antoine Delignat-Lavaud
Verified security for web applications,
started September 2012, supervised by Karthikeyan Bhargavan

PhD in progress: Jean Karim Zinzindohoue
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 in progress: Yannis Juglaret
Micro-policies and Secure Compilation,
started September 2015, supervised by Catalin Hritcu

10.2.3. Juries

Rémy Chrétien – Ph.D. – Jan. 11, 2016 – ENS Cachan
Automated analysis of equivalence properties for cryptographic protocols
Bruno Blanchet (reviewer)

Joeri de Ruiter – Ph.D. – Aug. 27, 2015 – Radboud University Nijmegen
Lesson learned in the analysis of the EMV and TLS security protocols
Karthikeyan Bhargavan (reviewer)

Bart van Delft – Licentiate – Mar. 14, 2015 – Chalmers University
Karthikeyan Bhargavan (discussion leader)

10.3. Popularization

10.3.1. Popular Press and Vulnerability Reports

The FREAK and Logjam attacks discovered by the miTLS team resulted in articles in many newspapers, magazine, and popular websites, including the New York Times, Wall Street Journal, Economist, and Wired. As a result of our discovery of these attacks and their popularization, major changes were made in all web browsers, web servers, and in the TLS protocol itself.

10.3.2. Seminars

- Catalin Hritcu: invited talks at CoqPL Workshop (Jan 2015), HP Labs France (May 2015), Inria Rennes (Jun 2015), PLAS Workshop (Jul 2015), Microsoft Research Redmond (Aug 2015), University of Washington (Aug 2015), ML Workshop (Sep 2015), Alcatel-Lucent Bell Labs (Dec 2015)
- Karthikeyan Bhargavan: invited talks at Real World Crypto (Jan 2015), Inria Nancy (Feb 2015), OSSIR Paris (Jun 2015), DefCon Crypto Village (Aug 2015), Catrel Workshop (Oct 2015)

11. Bibliography

Major publications by the team in recent years

[1] A. AZEVEDO DE AMORIM, M. DÉNÈS, N. GIANNARAKIS, C. HRITCU, B. C. PIERCE, A. SPECTOR-ZABUSKY, A. TOLMACH. *Micro-Policies: Formally Verified, Tag-Based Security Monitors*, in "IEEE Symposium on Security and Privacy (Oakland)", 2015, pp. 813–830

[2] G. BANA, H. COMON-LUNDH. *A Computationally Complete Symbolic Attacker for Equivalence Properties*, in "ACM Conference on Computer and Communications Security (CCS)", 2014, pp. 609–620

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

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

[5] K. BHARGAVAN, A. DELIGNAT-LAVAUD, S. MAFFEIS. *Language-Based Defenses Against Untrusted Browser Origins*, in "USENIX Security Symposium", 2013

[6] K. BHARGAVAN, C. FOURNET, M. KOHLWEISS, A. PIRONTI, P.-Y. STRUB. *Implementing TLS with Verified Cryptographic Security*, in "IEEE Symposium on Security and Privacy (Oakland)", 2013, pp. 445-462

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

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

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

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

Publications of the year

Articles in International Peer-Reviewed Journals

[11] D. CADÉ, B. BLANCHET. *Proved Generation of Implementations from Computationally Secure Protocol Specifications*, in "Journal of Computer Security", 2015, vol. 23, n^{o} 3, pp. 331-402, https://hal.inria.fr/hal-01102382

International Conferences with Proceedings

[12] D. ADRIAN, K. BHARGAVAN, Z. DURUMERIC, P. GAUDRY, M. GREEN, J. A. HALDERMAN, N. HENINGER, D. SPRINGALL, E. THOMÉ, L. VALENTA, B. VANDERSLOOT, E. WUSTROW, S. ZANELLA-BÉGUELIN, P. ZIMMERMANN. *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice*, in "ACM CCS 2015", Denver, Colorado, United States, 2015 ACM SIGSAC Conference on Computer and Communications Security, October 2015, 14 p. [*DOI :* 10.1145/2810103.2813707], https://hal.inria.fr/hal-01184171

[13] A. AZEVEDO DE AMORIM, M. DÉNÈS, N. GIANNARAKIS, C. HRITCU, B. C. PIERCE, A. SPECTOR-ZABUSKY, A. TOLMACH. *Micro-Policies: Formally Verified, Tag-Based Security Monitors*, in "2015 IEEE Symposium on Security and Privacy", San Jose, United States, 2015 IEEE Symposium on Security and Privacy, May 2015, pp. 813 - 830 [*DOI :* 10.1109/SP.2015.55], https://hal.inria.fr/hal-01265666

[14] B. BEURDOUCHE, K. BHARGAVAN, A. DELIGNAT-LAVAUD, C. FOURNET, M. KOHLWEISS, A. PIRONTI, P.-Y. STRUB, J. K. ZINZINDOHOUE. *A Messy State of the Union: Taming the Composite State Machines of TLS*, in "IEEE Symposium on Security & Privacy 2015", San Jose, United States, IEEE, May 2015, To appear, https://hal.inria.fr/hal-01114250

[15] K. BHARGAVAN, A. DELIGNAT-LAVAUD, A. PIRONTI. *Verified Contributive Channel Bindings for Compound Authentication*, in "Network and Distributed System Security Symposium (NDSS'15)", San Diego, United States, February 2015, To appear, https://hal.inria.fr/hal-01114248

[16] 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, https://hal.inria.fr/hal-01244855

[17] A. DELIGNAT-LAVAUD, K. BHARGAVAN. *Network-based Origin Confusion Attacks against HTTPS Virtual Hosting*, in "24th International Conference on World Wide Web", Florence, Italy, ACM, May 2015, To appear, https://hal.inria.fr/hal-01114246

[18] U. DHAWAN, C. HRITCU, R. RUBIN, N. VASILAKIS, S. CHIRICESCU, J. M. SMITH, J. T. F. KNIGHT, B. C. PIERCE, A. DEHON. *Architectural Support for Software-Defined Metadata Processing*, in "20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2015)", Istanbul, Turkey, ACM, March 2015, pp. 487-502 [*DOI :* 10.1145/2694344.2694383], https://hal.inria.fr/hal-01102378

[19] Z. PARASKEVOPOULOU, C. HRIŢCU, M. DÉNÈS, L. LAMPROPOULOS, B. C. PIERCE. *Foundational Property-Based Testing*, in "ITP 2015 - 6th conference on Interactive Theorem Proving", Nanjing, China, Lecture Notes in Computer Science, Springer, August 2015, vol. 9236 [*DOI :* 10.1007/978-3-319-22102-1_22], https://hal.inria.fr/hal-01162898

[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, https://hal.archives-ouvertes.fr/hal-01265793

Research Reports

[21] M. PAIOLA, B. BLANCHET. *From the Applied Pi Calculus to Horn Clauses for Protocols with Lists*, Inria, December 2015, n^{o} RR-8823, 45 p. , https://hal.inria.fr/hal-01239290

References in notes

[22] M. ABADI, B. BLANCHET. *Analyzing Security Protocols with Secrecy Types and Logic Programs*, in "Journal of the ACM", January 2005, vol. 52, n^{o} 1, pp. 102–146

[23] 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^{o} 3, pp. 1–59

[24] 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

[25] A. AZEVEDO DE AMORIM, N. COLLINS, A. DEHON, D. DEMANGE, C. HRITCU, D. PICHARDIE, B. C. PIERCE, R. POLLACK, A. TOLMACH. *A Verified Information-Flow Architecture*, September 2015, arXiv:1509.06503; Submitted to special issue of the Journal of Computer Security (JCS) on Verified Information Flow Security

[26] A. AZEVEDO DE AMORIM, M. DÉNÈS, N. GIANNARAKIS, C. HRITCU, B. C. PIERCE, A. SPECTOR-ZABUSKY, A. TOLMACH. *Micro-Policies: Formally Verified, Tag-Based Security Monitors*, in "36th IEEE Symposium on Security and Privacy (Oakland S&P)", IEEE Computer Society, May 2015, pp. 813–830

[27] G. BANA, H. COMON-LUNDH. *A Computationally Complete Symbolic Attacker for Equivalence Properties*, in "2014 ACM SIGSAC Conference on Computer and Communications Security", Scottsdale, United States, ACM, November 2014, pp. 609-620

[28] G. BANA, K. HASEBE, M. OKADA. *Computationally Complete Symbolic Attacker and Key Exchange*, in "ACM Conference on Computer and Communications Security (CCS'13)", Berlin, Germany, ACM, 2013, pp. 1231–1246

[29] 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^{o} 2, 8 p.

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

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

[32] 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

[33] 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

[34] B. BLANCHET, M. ABADI, C. FOURNET. *Automated Verification of Selected Equivalences for Security Protocols*, in "Journal of Logic and Algebraic Programming", February–March 2008, vol. 75, n^{o} 1, pp. 3–51

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

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

[37] B. BLANCHET, M. PAIOLA. *Automatic Verification of Protocols with Lists of Unbounded Length*, in "ACM conference on Computer and communications security (CCS'13)", Berlin, Germany, ACM, November 2013, pp. 573–584

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

[39] T. CHOTHIA, B. SMYTH, C. STAITE. *Automatically Checking Commitment Protocols in ProVerif without False Attacks*, in "POST'15: 4th Conference on Principles of Security and Trust", LNCS, Springer, 2015, vol. 9036

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

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

[42] 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

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

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

[45] Y. JUGLARET, C. HRITCU, A. AZEVEDO DE AMORIM, B. C. PIERCE, A. SPECTOR-ZABUSKY, A. TOLMACH. *Towards a Fully Abstract Compiler Using Micro-Policies: Secure Compilation for Mutually Distrustful Components*, October 2015, Technical Report, arXiv:1510.00697

[46] L. LAMPROPOULOS, B. C. PIERCE, C. HRITCU, J. HUGHES, Z. PARASKEVOPOULOU, LI-YAO. XIA. *Making Our Own Luck: A Language For Random Generators*, July 2015, Draft

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

[48] E. A. QUAGLIA, B. SMYTH. *Constructing secret, verifiable auction schemes from election schemes*, 2015

[49] B. SMYTH, S. FRINK, M. R. CLARKSON. *Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ*, 2015

[50] B. SMYTH, M. D. RYAN, L. CHEN. *Formal analysis of privacy in Direct Anonymous Attestation schemes*, in "Science of Computer Programming", 2015, vol. 111, n$^{\text{o}}$ 2

[51] B. SMYTH. *Secrecy and independence for election schemes*, 2015

[52] 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

[53] 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$^{\text{o}}$ 4, pp. 402-451

[54] 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