Activity Report 2013

Project-Team PROSECCO

Programming securely with cryptography

RESEARCH CENTER
Paris - Rocquencourt

THEME
Security and Confidentiality

Table of contents

Project-Team PROSECCO

Keywords: Programming Languages, Security, Formal Methods, Cryptographic Protocols, Automated Verification

Creation of the Project-Team: 2012 July 01.

1. Members

Research Scientists
Karthikeyan Bhargavan [Inria, Team leader, HdR]
Bruno Blanchet [Inria, Senior Researcher, HdR]
Graham Steel [Inria, Researcher, HdR]
Catalin Hritcu [Inria, Researcher from Oct 2013]
Alfredo Pironti [Inria, Starting Research position from Oct 2013]

Engineer
Romain Bardou [Inria]

PhD Students
Evmorfia-Iro Bartzia [Inria]
David Cadé [Inria]
Antoine Delignat-Lavaud [Inria]
Robert Künnemann [Inria]
Miriam Paiola [Inria]

Post-Doctoral Fellows
Gergely Bana [Inria]
Benjamin Smyth [Inria]

Visiting Scientist
Pierre-Malo Deniélou [Lecturer, Royal Holloway, University of London]

Administrative Assistants
Stephanie Aubin [Inria, until Aug 2013]
Anna Bednarik [Inria, from Oct 2013]

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 new team PROSECCO is to draw upon our expertise in security 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 an IDE. 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.

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 a static and dynamic verification tool for client-side JavaScript web applications, annotated JML libraries for verifying the security of Android smartphone applications, and the type systems F7 and F* for verifying clients and servers written in F#. In addition, we will extend our security API tools to analyse the secure elements and cryptographic roots-of-trust embedded in new-generation smartphones. We plan to combine these tools to analyze the security of multi-party web applications, consisting of clients on browsers and smartphones, and servers in the cloud.

2.2. Highlights of the Year

This year, we published 6 articles in international journals and 11 articles in peer-reviewed international conferences, including presitigious conferences such as IEEE S&P (1), ACM CCS (2), Usenix Security (1), ESORICS (2), and POST (3). In addition to these, we published 1 PhD thesis and several technical reports. We also have 3 articles already accepted for publication in international conferences in 2013: POPL (2), NDSS (1).

We released updates to several verification tools and software packages. We discovered and reported major security vulnerabilites in dozens of commercial software packages, hardware devices, and websites. The work of our group also spun-off a new startup company created by Graham Steel, and we continue to collaborate with this startup.

Of our work published in 2013, we would like to highlight the following:

- Our paper in IEEE S&P 2013 [21] presents the first cryptographically verified implementation of TLS.
- Our work on the computational analysis of cryptographic protocols yielded new results and major publications [19], [26].
- Our work on formally analyzing web application security uncovered major attacks on browsers and websites and proposed novel language-based verified solutions [20], [29], [25].

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 (in 2003, 2008, 2009, and 2011) 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 [49]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [47] 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* [43]. 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 [41], [34], correspondences (including authentication) [42], and observational equivalences [40]. 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 [37], JFK [35], and Web Services Security [39]. 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 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 [45], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [46]. 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

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# [37]. 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 [36] 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 [38]. F* [51] 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.

An alternative approach is to directly verify cryptographic applications in the computational model by typing. A recent work [48] 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.

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. 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 F7 to verify their correctness.

4. Application Domains

4.1. Cryptographic protocol implementations

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, 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 authenticate authorize users using a single sign-on protocol such as OAuth, a cloud storage service may encrypt user files on the server-side using XML encryption, and a password manager may encrypt passwords in the browser using a JavaScript cryptographic library. We build verification tools that can analyze such usages in commercial web applicaitons and evaluate their security against sophisticated web-based attacks.

5. Software and Platforms

5.1. ProVerif

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

PROVERIF (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 proverif.inria.fr, under the GPL license.

5.2. CryptoVerif

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

CRYPTOVERIF(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 cryptoverif.inria.fr, under the CeCILL license.

5.3. Cryptosense Analyzer

Participants: Graham Steel [correspondant], Romain Bardou.

See also the web page http://cryptosense.com.

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

In June 2013 we submitted a patent application (13 55374) on the reverse engineering process. We also concluded a license agreement between Inria PROSECCO and the nascent spin-off company Cryptosense to commercialize the tool.

5.4. miTLS

Participants: Alfredo Pironti [correspondant], Karthikeyan Bhargavan, Cedric Fournet [Microsoft Research], Pierre-Yves Strub [IMDEA], Markulf Kohlweiss [Microsoft Research].

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, and two updates to the software were released in 2013. The software and associated research materials are available from http://mitls.rocq.inria.fr.

5.5. WebSpi

Participants: Karthikeyan Bhargavan [correspondant], Sergio Maffeis [Imperial College London], Chetan Bansal [BITS Pilani-Goa], Antoine Delignat-Lavaud.

WebSpi is a library that aims to make it easy to develop models of web security mechanisms and protocols and verify them using ProVerif. It captures common modeling idioms (such as principals and dynamic compromise) and defines a customizable attacker model using a set of flags. It defines an attacker API that is designed to make it easy to extract concrete attacks from ProVerif counterexamples.

WebSpi has been used to analyze social sign-on and social sharing services offered by prominent social networks, such as Facebook, Twitter, and Google, on the basis of new open standards such as the OAuth 2.0 authorization protocol.

WebSpi has also been used to investigate the security of a number of cryptographi web applications, including password managers, cloud storage providers, an e-voting website and a conference management system.

WebSpi is under development and released as an open source library at http://prosecco.inria.fr/webspi/

5.6. Defensive JavaScript

Participants: Antoine Delignat-Lavaud [correspondant], Karthikeyan Bhargavan, Sergio Maffeis [Imperial College London].

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. DJS is especially useful to write security APIs that can be loaded in untrusted pages, for instance an OAuth library such as the one used by "Login with Facebook". It is also useful to write secure host-proof web applications, and more generally for cryptography that happens on the browser.

The DJS type checker and various libraries written in DJS are available from http://www.defensivejs.com.

6. New Results

6.1. Verification of Security Protocols with Lists in the Symbolic Model

Participants: Bruno Blanchet, Miriam Paiola.

The symbolic model of protocols, or Dolev-Yao model is an abstract model in which messages are represented by terms. Our protocol verifier PROVERIF relies on this model. This year, we have mainly worked on the verification of protocols with lists in this model.

We designed a novel automatic technique for proving secrecy and authentication properties for security protocols that manipulate lists of unbounded length, for an unbounded number of sessions. This result is achieved by extending the Horn clause approach of the automatic protocol verifier ProVerif. We extended the Horn clauses to be able to represent lists of unbounded length. We adapted the resolution algorithm to handle the new class of Horn clauses, and proved the soundness of this new algorithm. We have implemented our algorithm and successfully tested it on several protocol examples, including XML protocols coming from web services. This work has been published in [22] and our prototype is available at http://prosecco.inria.fr/personal/bblanche/publications/BlanchetPaiolaCCS13.html.

Last year, we published a conference paper that shows that, for a limited class of protocols, if a protocol is proven secure by ProVerif with lists of length one, then it is secure for lists of unbounded length. A journal version [50] of this paper has now been accepted.

6.2. Generation of Implementations Proved Secure in the Computational model

Participants: Bruno Blanchet, David Cadé.

The computational model of protocols considers messages as bitstrings, which is more realistic than the formal model, but also makes the proofs more difficult. Our verifier CRYPTOVERIF is sound in this model. This year, we have continued working on our compiler from CRYPTOVERIF speficications to OCaml. Using CryptoVerif and this compiler, we can prove security properties of specifications of protocols in the computational model and generate runnable implementations from such proved specifications. We have published a journal paper on our implementation of SSH generated using this compiler [13] and a proof that this compiler preserves security [23], and we have submitted a journal version of this proof. David Cadé also defended his PhD thesis on this topic [44].

6.3. Computationally Complete Symbolic Attacker and Key Exchange

Participants: Gergely Bana [correspondant], Koji Hasebe, Mitsuhiro Okada.

Around year 2000, various research groups started looking into the relevance of symbolic verification techniques to computational security. If a symbolic verification technique results computational guarantees, we say that computational soundness holds. One of the major concerns has been that the usual Dolev-Yao symbolic attacker that automated symbolic tools used exclusively (to search for attacks) at that time did not seem to allow satisfactory soundness results, only with serious limitations. One possible promising approach to overcome this problem is to derive security guarantees directly as CryptoVerif or F7 does. As an alternative approach, in 2012, Bana and Comon-Lundh introduced a notion they called computationally complete symbolic attacker. With this technique, elimination of the possibility of a computational attack would also mean that computational attack does not exist without the limitations that the Dolev-Yao technique required. Their symbolic attacker can do everything that is not forbidden by conditions derived from standard computational assumptions on the primitives. In this current work, based on predicates for "key compromise", we provided such conditions to handle secure encryption even keys are allowed to be sent. We examined both IND-CCA2 and KDM-CCA2 encryptions, both symmetric and asymmetric situations as well as INT-CTXT encryptions. We verified (by hand) a number of protocols as the symmetric Needham-Schroeder protocol, Otway-Rees protocol, Needham-Schroeder-Lowe protocol. Furthermore, we also made some improvements in the computational semantics, and have established a relationship between the computational semantics of Bana and Comon-Lundh and Fitting's embedding of classical logic into S4. This work was published at CCS'13 [19].

6.4. Formal Models and Concrete Attacks on Web Applications

Participants: Karthikeyan Bhargavan [correspondant], Sergio Maffeis, Chetan Bansal, Antoine Delignat-Lavaud, Michael May.

Modern web applications are built as a combination of mostly static servers that host user data and highly dynamic client-side applications that process and present the data to the user. These client-side applications may be hosted as JavaScript within a browser or within custom applications written, say, for smartphones. Hence, in addition to traditional server-side mechanisms, the security of these applications increasingly depends on the correct use of browser-based security mechanisms, client-side access control, and cryptography. These mechanisms are often new, ad hoc, and deserving of close analysis.

Our approach is to formally model various client- and server-side security mechanisms for web applications and rigorously analyze their real-world deployments. When our formal analyses find attacks, we test them against example web applications, report vulnerabilities to various vendors, design countermeasures, and use automated security protocol analysis tools formally verify that our countermeasure resists a large class of attacks. This year, we published three papers in this area. At ESSoS, we formally modeled the authorization policies of common Android apps, found new attacks, and proposed a verified authorization framework [27]. At POST, we formally modeled various cloud-based encrypted storage applications and found both cryptographic and web attacks on them, resulting in patches to these websites and novel countermeasures [20]. At Usenix Security, we proposed a new, safer language for security-critical web components [25]. Defensive JavaScript is a subset of JavaScript that guarantees isolation from other (potentially untrusted) scripts on the same page. This enables, for the first time, the design of cryptographic and single sign-on components that can be formally guaranteed to preserve its secrets even if the hosting website is subject to a cross-site scripting attack.

6.5. Attacks and Proofs for TLS Implementations

Participants: Alfredo Pironti [correspondant], Karthikeyan Bhargavan, Pierre-Yves Strub, Cedric Fournet, Markulf Kohlweiss, Antoine Delignat-Lavaud.

TLS is possibly the most used secure communications protocol, with a 18-year history of flaws and fixes, ranging from its protocol logic to its cryptographic design, and from the Internet standard to its diverse implementations. We have been engaged in a long-term project on verifying TLS implementations and this project is now coming to fruition, with a number of papers are now in the pipeline. We present the main published results below, other papers have been submitted for review.

We have developed a verified reference implementation of TLS 1.2, called miTLS. Our code fully supports its wire formats, ciphersuites, sessions and connections, re-handshakes and resumptions, alerts and errors, and data fragmen- tation, as prescribed in the RFCs; it interoperates with mainstream web browsers and servers. At the same time, our code is carefully structured to enable its modular, automated verification, from its main API down to computational assumptions on its cryptographic algorithms. Our implementation is written in F# and specified in F7. We present security specifications for its main components, such as authenticated stream encryption for the record layer and key establishment for the handshake. We describe their verification using the F7 refinement typechecker. To this end, we equip each cryptographic primitive and construction of TLS with a new typed interface that captures its security prop- erties, and we gradually replace concrete implementations with ideal functionalities. We finally typecheck the protocol state machine, and thus obtain precise security theorems for TLS, as it is implemented and deployed. We also revisit classic attacks and report a few new ones. This work was published at IEEE S&P 2013 [21].

In parallel with this long-term constructive project, we have been analyzing the use of TLS in existing web applications, and our analyses uncovered a number of attacks, leading to patched in popular browsers like Chrome, Internet Explorer, and Firefox, as well as websites like Google and Akamai.

One of these classes of attacks was published at WOOT'13 [29]. In this paper, we identify logical web application flaws which can be exploited by TLS truncation attacks to desynchronize the user- and server-perspective of an application's state. It follows immediately that servers may make false assumptions about users, hence, the flaw constitutes a security vulnerability. Moreover, in the context of authentication systems, we exploit the vulnerability to launch the following practical attacks: we exploit the Helios electronic voting system to cast votes on behalf of honest voters, take full control of Microsoft Live accounts, and gain temporary access to Google accounts.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

Contract with Airbus (http://www.airbus.com/), on the modeling and verification of avionic security protocols. Participant: Bruno Blanchet. From October to December 2013.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. 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.

8.1.2. FUI

8.1.2.1. 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

8.2. European Initiatives

8.2.1. FP7 Projects

8.2.1.1. CRYSP

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

Type: IDEAS ()

Instrument: ERC Starting Grant (Starting)

Duration: November 2010 - October 2015

Coordinator: Karthikeyan Bhargavan, Inria (France)

Abstract: The goal of this grant is to develop a collaborative specification framework and to build incremental, modular, scalable verification techniques that enable a group of collaborating programmers to build an application and its security proof side-by-side. We propose to validate this framework by developing the first large-scale web application and full-featured cryptographic protocol libraries with formal proofs of security.

8.3. International Initiatives

8.3.1. Inria International Partners

8.3.1.1. Informal International Partners

- K. Bhargavan, A. Pironti, and A. Delignat-Lavaud work closely with Microsoft Research in Cambridge, Redmond, Silicon Valley, and Bangalore (C. Fournet, N. Swamy, M. Abadi, P. Naldurg)
- G. Steel and R. Bardou work closely with University of Venice, Italy (R. Foccardi).
- G. Bana works closely with Keio University Japan
- E-I. Bartzia works closely with IMDEA Madrid (P-Y. Strub)

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Pierre-Malo Denielou (Lecturer, Royal Holloway, University of London) visited us for two months as professeur invité.
- Sergio Maffeis (Imperial College, London) visited us as part of an ongoing collaboration.
- Cédric Fournet (Researcher, Microsoft Researcher) visited us as part of an ongoing collaboration.

8.4.2. Visits to International Teams

- Alfredo Pironti visited Microsoft Research Cambridge (UK) several times, as part of a long-term collaboration
- Gergely Bana visited Keio University (Japan), ICT Lisbok (Portugal), and Queen Mary University of London in Nov 2013

- Benjamin Smyth visited Toshiba, Japan and Univesity of Birmingham (UK)

9. Dissemination

9.1. Scientific Animation

9.1.1. Journal Editorial Boards

Associate Editor

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

9.1.2. Conference Program Committees

- FCS – June 2013, New Orleans, LA, USA: Bruno Blanchet (PC co-chair)
- POST – April 2014, Grenoble, France: Bruno Blanchet
- SEC@SAC – March 2013, Coimbra, Portugal: Graham Steel (PC co-chair)
- CSF – June 2013, New Orleans, USA: Graham Steel
- IWIL – December 2013, Stellenbosch, South Africa: Graham Steel

9.1.3. Participation to Workshops and Conferences

ETAPS – March 2013, Rome, Italy: Bruno Blanchet, David Cadé, Miriam Paiola, Benjamin Smyth

CSF, FCS, FCC – June 2013, New Orleans, LA, USA: Bruno Blanchet, Benjamin Smyth

CCS – November 2013, Berlin, Germany: Bruno Blanchet, David Cadé, Miriam Paiola, Gergely Bana

S&P – May 2013, San Francisco, USA: Karthikeyan Bhargavan, Alfredo Pironti

Usenix Security, WOOT – August 2013, Washington DC, USA: Karthikeyan Bhargavan, Alfredo Pironti, Antoine Delignat-Lavaud

CryptoForma – September 2013, San Francisco, USA: Alfredo Pironti

Black Hat USA – July 2013, Las Vegas, USA: Benjamin Smyth

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

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

Karthikeyan Bhargavan, TDs in INF431 and INF321, introductory programming language courses at Ecole Polytechnique, France

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

Bruno Blanchet, Automatic Verification of Cryptographic Protocols in the Symbolic Model, the Automatic Verifier ProVerif, 6h equivalent TD, 13th International School on Foundations of Security Analysis and Design (FOSAD'13), Bertinoro, Italy

Miriam Paiola, Internet et Outils - IO2, TP, 52h equivalent TD, L1 en Math-Info, Université Paris VII, France

Miriam Paiola, Automates finis - AF4, TP, 26h equivalent TD, L2 en Informatique, Université Paris VII, France

Graham Steel, Formal Methods for the Science of Security Summer School, UIUC, July 2013, "Security APIs" (4 hours teaching)

Graham Steel, SecAppDev Days Training, Katholieke Universiteit Leuven, March 2013, "Security APIs" (3 hours teaching)

Graham Steel, University of Venice Ca' Foscari, PhD course "Security APIs" October/November 2013, (20 hours teaching)

Evmorfia-Iro Bartzia, Introduction to Mathematical Cryptography (cours + TD), MACS3: Mathématiques appliquées et calcul scientifique 3eme année, Université Paris 13

Evmorfia-Iro Bartzia, Elliptic Curves and Complex Torus (TD), MFPI: Mathématiques Fondamentales et Protection de l'Information, Université Paris 8

9.2.2. Supervision

- David Cadé
  Proved Implementations of Cryptographic Protocols in the Computational Model,
  defended on December 16, 2013, supervised by Bruno Blanchet
- Miriam Paiola
  Automatic Verification of Group Protocols,
  since November 2010, supervised by Bruno Blanchet
- Robert Künnemann, *Secure APIs and Simulation-Based Security*, Started Oct. 2010, supervised by Steve Kremer (CASSIS) and Graham Steel, submitted October 2013 will defend January 2014
- Gavin Keighren, *A Type System for Security APIs*, since 2007 (submitted August 2013), advisors Graham Steel and David Aspinall (University of Edinburgh). Will defend January 2014.
- Antoine Delignat-Lavaud, since 2012, supervised by Karthikeyan Bhargavan
- Evmorfia-Iro Bartzia, since 2011, supervised by Karthikeyan Bhargavan and Pierre-Yves Strub

9.2.3. Internships

- Benjamin Beurdouche did an undergraduate internship under the supervision of Alfredo Pironti and Karthikeyan Bhargavan
- Adam McCarthy did a masters internship under the supervision of Benjamin Smyth

9.2.4. Juries

- Jannik Dreier – Ph.D. – Nov. 25, 2013 – Université de Grenoble
  Bruno Blanchet
- Maria Christofy – Ph.D. –Feb. 15, 2013– UVSQ
  Graham Steel (rapporteur)
- François Dupressoir – Ph.D.–Feb. 6, 2013– Open University
  Graham Steel (rapporteur)

9.3. Popularization

9.3.1. Vulnerability Reports

- Benjamin Smyth and Alfredo Pironti reported TLS truncation vulnerabilities to Helios, Microsoft, and Google, and presented a talk at BlackHat USA. They were acknowledged in Google's Hall of Fame.
- Karthikeyan Bhargavan and Antoine Delignat-Lavaud reported HTTPS header truncation vulnerabilities in Google Chrome and Apple Safari, resulting in a security update to Google Chrome.
- Karthikeyan Bhargavan and Antoine Delignat-Lavaud reported TLS protocol-level vulnerabilities in Internet Explorer, Google Chrome, and Mozilla Firefox, resulting in security updates to all three.
- Antoine Delignat-Lavaud reported new vulnerabilites in Akamai-hosted websites, resulting in an update to Akamai's web caching network.

10. Bibliography

Major publications by the team in recent years

[1] 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

[2] 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, http://hal.inria.fr/hal-00918848

[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, S. MAFFEIS. *Language-Based Defenses Against Untrusted Browser Origins*, in "Proceedings of the 22th USENIX Security Symposium", 2013, http://hal.inria.fr/hal-00863372

[5] K. BHARGAVAN, C. FOURNET, A. D. GORDON. *Modular verification of security protocol code by typing*, in "37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'10)", 2010, pp. 445-456

[6] K. BHARGAVAN, C. FOURNET, M. KOHLWEISS, A. PIRONTI, P.-Y. STRUB. *Implementing TLS with Verified Cryptographic Security*, in "IEEE Symposium on Security & Privacy (Oakland)", San Francisco, United States, 2013, pp. 445-462, http://hal.inria.fr/hal-00863373

[7] B. BLANCHET, M. PAIOLA. *Automatic Verification of Protocols with Lists of Unbounded Length*, in "CCS'13 - ACM Conference on Computer and Communications Security", Berlin, Germany, ACM, 2013, pp. 573–584 [*DOI :* 10.1145/2508859.2516679], http://hal.inria.fr/hal-00918849

[8] B. BLANCHET, D. POINTCHEVAL. *Automated Security Proofs with Sequences of Games*, in "CRYPTO'06", Santa Barbara, CA, C. DWORK (editor), LNCS, Springer Verlag, August 2006, vol. 4117, pp. 537–554

[9] M. BORTOLOZZO, M. CENTENARO, R. FOCARDI, G. STEEL. *Attacking and Fixing PKCS#11 Security Tokens*, in "Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)", Chicago, Illinois, USA, ACM Press, October 2010, pp. 260-269 [*DOI :* 10.1145/1866307.1866337], http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BCFS-ccs10.pdf

[10] V. CORTIER, G. STEEL, C. WIEDLING. *Revoke and let live: a secure key revocation api for cryptographic devices*, in "ACM Conference on Computer and Communications Security (CCS'12)", 2012, pp. 918-928

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] R. KÜNNEMANN. , *Foundations for analyzing security APIs in the symbolic and computational model*, École normale supérieure de Cachan - ENS Cachan, January 2014, http://hal.inria.fr/tel-00942459

Articles in International Peer-Reviewed Journals

[12] M. BACKES, C. HRITCU, M. MAFFEI. *Union, Intersection, and Refinement Types and Reasoning About Type Disjointness for Secure Protocol Implementations*, in "Special issue of the Journal of Computer Security (JCS) for TOSCA-SecCo", 2013, Accepted for publication, http://hal.inria.fr/hal-00918846

[13] D. CADÉ, B. BLANCHET. *From Computationally-Proved Protocol Specifications to Implementations and Application to SSH*, in "Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)", 2013, vol. 4, n^{o} 1, pp. 4–31, Special issue ARES'12, http://hal.inria.fr/hal-00863374

[14] V. CORTIER, B. SMYTH. *Attacking and fixing Helios: An analysis of ballot secrecy*, in "Journal of Computer Security", 2013, vol. 21, n^{o} 1, pp. 89–148 [*DOI :* 10.3233/JCS-2012-0458], http://hal.inria.fr/hal-00940324

[15] V. CORTIER, G. STEEL. *A Generic Security API for Symmetric Key Management on Cryptographic Devices*, in "Information and Computation", 2013, To appear, http://hal.inria.fr/hal-00881072

[16] M. PAIOLA, B. BLANCHET. *Verification of Security Protocols with Lists: from Length One to Unbounded Length*, in "Journal of Computer Security", 2013, vol. 21, n^{o} 6, pp. 781–816, Special issue POST'12, http://hal.inria.fr/hal-00939187

[17] N. SWAMY, J. CHEN, C. FOURNET, P.-Y. STRUB, K. BHARGAVAN, J. YANG. *Secure distributed programming with value-dependent types*, in "J. Funct. Program.", 2013, vol. 23, n^{o} 4, pp. 402-451, http://hal.inria.fr/hal-00939188

International Conferences with Proceedings

[18] A. A. AMORIM, N. COLLINS, A. DEHON, D. DEMANGE, C. HRITCU, D. PICHARDIE, B. C. PIERCE, R. POLLACK, A. TOLMACH. *A Verified Information-Flow Architecture*, in "41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", San Diego, CA, United States, 2014, To appear [*DOI :* 10.1145/2535838.2535839], http://hal.inria.fr/hal-00918847

[19] 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, http://hal.inria.fr/hal-00918848

[20] C. BANSAL, K. BHARGAVAN, A. DELIGNAT-LAVAUD, S. MAFFEIS. *Keys to the Cloud: Formal Analysis and Concrete Attacks on Encrypted Web Storage*, in "2nd Conference on Principles of Security and Trust (POST 2013)", Rome, Italy, D. BASIN, J. MITCHELL (editors), lncs, spv, 2013, vol. 7796, pp. 126–146, http://hal.inria.fr/hal-00863375

[21] K. BHARGAVAN, C. FOURNET, M. KOHLWEISS, A. PIRONTI, P.-Y. STRUB. *Implementing TLS with Verified Cryptographic Security*, in "IEEE Symposium on Security & Privacy (Oakland)", San Francisco, United States, 2013, pp. 445-462, http://hal.inria.fr/hal-00863373

[22] B. BLANCHET, M. PAIOLA. *Automatic Verification of Protocols with Lists of Unbounded Length*, in "CCS'13 - ACM Conference on Computer and Communications Security", Berlin, Germany, ACM, 2013, pp. 573–584 [*DOI :* 10.1145/2508859.2516679], http://hal.inria.fr/hal-00918849

[23] D. CADÉ, B. BLANCHET. *Proved Generation of Implementations from Computationally-Secure Protocol Specifications*, in "2nd Conference on Principles of Security and Trust (POST 2013)", Rome, Italy, D. BASIN, J. MITCHELL (editors), lncs, spv, 2013, vol. 7796, pp. 63–82, http://hal.inria.fr/hal-00863376

[24] V. CHEVAL, B. BLANCHET. *Proving More Observational Equivalences with ProVerif*, in "2nd Conference on Principles of Security and Trust (POST 2013)", Rome, Italy, D. BASIN, J. MITCHELL (editors), lncs, spv, 2013, vol. 7796, pp. 226–246, http://hal.inria.fr/hal-00863377

[25] A. DELIGNAT-LAVAUD, K. BHARGAVAN, S. MAFFEIS. *Language-Based Defenses Against Untrusted Browser Origins*, in "Proceedings of the 22th USENIX Security Symposium", 2013, http://hal.inria.fr/hal-00863372

[26] S. KREMER, R. KÜNNEMANN, G. STEEL. *Universally Composable Key-Management*, in "18th European Symposium on Research in Computer Security (ESORICS'13)", Egham, United Kingdom, J. CRAMPTON, S. JAJODIA (editors), Lecture Notes in Computer Science, Springer, 2013, vol. 8134 [*DOI :* 10.1007/978-3-642-40203-6_19], http://hal.inria.fr/hal-00878632

[27] M. J. MAY, K. BHARGAVAN. *Towards Unified Authorization for Android*, in "5th International Symposium on Engineering Secure Software and Systems (ESSoS 2013)", llncs, spv, 2013, vol. 7781, pp. 42-57, http://hal.inria.fr/hal-00863384

[28] B. SMYTH, D. BERNHARD. *Ballot secrecy and ballot independence coincide*, in "ESORICS'13: 18th European Symposium on Research in Computer Security", Egham, United Kingdom, LNCS, Springer, 2013, vol. 8134, pp. 463-480, http://hal.inria.fr/hal-00863370

[29] B. SMYTH, A. PIRONTI. *Truncating TLS Connections to Violate Beliefs in Web Applications*, in "WOOT'13: 7th USENIX Workshop on Offensive Technologies", Washington, United States, USENIX Association, 2013, First appeared at Black Hat USA 2013, http://hal.inria.fr/hal-00863371

[30] N. SWAMY, C. FOURNET, A. RASTOGI, K. BHARGAVAN, J. CHEN, P.-Y. STRUB, G. BIERMAN. *Gradual Typing Embedded Securely in JavaScript*, in "POPL 2014 - 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", San Diego, CA, United States, ACM, 2014, pp. 425-437 [*DOI :* 10.1145/2535838.2535889], http://hal.inria.fr/hal-00940836

Research Reports

[31] C. BANSAL, K. BHARGAVAN, A. DELIGNAT-LAVAUD, S. MAFFEIS. , *Discovering Concrete Attacks on Website Authorization by Formal Analysis*, Inria, April 2013, n° RR-8287, 46 p. , http://hal.inria.fr/hal-00815834

[32] M. DAUBIGNARD, D. LUBICZ, G. STEEL. , *A Secure Key Management Interface with Asymmetric Cryptography*, Inria, 2013, n° RR-8274, http://hal.inria.fr/hal-00805987

[33] A. DELIGNAT-LAVAUD, K. BHARGAVAN, S. MAFFEIS. , *Embedding of Security Components in Untrusted Third-Party Websites*, Inria, April 2013, n° RR-8285, 32 p. , http://hal.inria.fr/hal-00815800

References in notes

[34] 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

[35] 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

[36] 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.

[37] 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.

[38] 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

[39] 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

[40] 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

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

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

[43] 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

[44] D. CADÉ. , *Proved Implementations of Cryptographic Protocols in the Computational Model*, Université Paris VII, December 2013

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

[46] 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

[47] 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

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

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

[50] M. PAIOLA, B. BLANCHET. *Verification of Security Protocols with Lists: from Length One to Unbounded Length*, in "Journal of Computer Security", December 2013, vol. 21, n^{o} 6, pp. 781–816, Special issue POST'12

[51] 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