Activity Report 2012

Project-Team PROSECCO

Programming Securely with Cryptography

RESEARCH CENTER
Paris - Rocquencourt

THEME
Programs, Verification and Proofs

Table of contents

1. Members **1**
2. Overall Objectives **1**
2.1. Programming securely with cryptography 1
2.1.1. Symbolic verification of cryptographic applications 2
2.1.2. Computational verification of cryptographic applications 2
2.1.3. Provably secure web applications 2
2.2. Highlights of the Year 2
3. Scientific Foundations **3**
3.1. Symbolic verification of cryptographic applications 3
3.1.1. Verifying cryptographic protocols with ProVerif 3
3.1.2. Verifying security APIs using Tookan 3
3.1.3. Verifying cryptographic applications using F7 4
3.2. Computational verification of cryptographic applications 4
3.3. Provably secure web applications 5
4. Application Domains **5**
4.1. Cryptographic protocol implementations 5
4.2. Hardware-based security APIs 5
4.3. Web application security 6
5. Software **6**
5.1. ProVerif 6
5.2. CryptoVerif 6
5.3. Tookan 7
5.4. miTLS 7
5.5. WebSpi 7
5.6. Defensive JavaScript 7
6. New Results **8**
6.1. Verification of Security Protocols in the Symbolic Model 8
6.1.1. Verification of Protocols with Lists 8
6.1.2. Proving More Process Equivalences with ProVerif 8
6.2. Verification of Security Protocols in the Computational Model 8
6.2.1. Generation of Implementations Proved Secure in the Computational model 8
6.2.2. Proof of One-Encryption Key Exchange using CryptoVerif 9
6.3. New Attacks on RSA PKCS#1 v1.5 9
6.4. Security Proofs for Revocation 9
6.5. Discovering Concrete Attacks on Web Applications by Formal Analysis 9
6.6. Attacks and Proofs for TLS Implementations 10
7. Bilateral Contracts and Grants with Industry **11**
8. Partnerships and Cooperations **11**
8.1. National Initiatives 11
8.2. European Initiatives 11
8.3. International Initiatives 11
8.4. International Research Visitors 12
8.4.1. Visits of International Scientists 12
8.4.2. Visits to International Teams 12
9. Dissemination **12**
9.1. Editorial Boards 12
9.2. Organizers 12
9.3. Program Committees 12
9.4. Vulnerability Reports 13

9.5. Teaching 13
9.6. Ph.D in progress 13
9.7. Ph.D/Habilitation Committees 13
9.8. Invited Talks 13
9.9. Participation to Workshops and Conferences 14
10. Bibliography **14**

Project-Team PROSECCO

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

Creation of the Project-Team: July 01, 2012 .

1. Members

Research Scientists
Karthikeyan Bhargavan [Team leader, HdR]
Bruno Blanchet [Senior Researcher, HdR]
Graham Steel [Researcher, HdR]

Engineer
Romain Bardou [Engineer]

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

Post-Doctoral Fellows
Alfredo Pironti
Ben Smyth
Gergei Bana [MSR-Inria Joint Centre]

Visiting Scientist
Michael May [Faculty Lecturer, Kinneret College on the Sea of Galilee, Israel]

Administrative Assistant
Stephanie Aubin [shared with another team]

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 5 articles in international journals and 11 articles in peer-reviewed international conferences, including presitigious conferences such as CCS (1), CRYPTO (1), and CSF (2). In addition to these, we published 1 HDR thesis, 3 master's theses, 4 technical reports, and 5 workshop papers. We also have 4 articles already accepted for publication in international conferences in 2013.

We released updates to 3 verification tools and released 3 new software packages. We discovered and reported major security vulnerabilites in dozens of commercial software packages, hardware devices, and websites.

Of our work published in 2012, we would like to highlight the following:

- Our paper in CRYPTO 2012 [22] describing new attacks on cryptographic hardware devices, which got significant interest from both the cryptographer community and from the press.
- Our work on generating implementation code from verified models of cryptographic protocols [26], [27].
- Our work on formally analyzing web application security using automated verification tools, which uncovered major attacks in popular websites and web browsers [21], [24], [20].

3. Scientific Foundations

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 [52]; it has motivated a serious research effort on the formal analysis of cryptographic protocols, starting with [50] 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* [46]. 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 [44], [38], correspondences (including authentication) [45], and observational equivalences [43]. 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 [13], JFK [39], and Web Services Security [42]. 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 [48], results which were extended by formal analysis work using a Dolev-Yao style model of the standard [49]. 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# [13]. 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 [40] 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 [41]. F* [53] 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 [51] 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

5.1. ProVerif

Participants: Bruno Blanchet [correspondant], Xavier Allamigeon [April–July 2004], Vincent Cheval [Sept. 2011–], Ben 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. Tookan

Participants: Graham Steel [correspondant], Romain Bardou.

See also the web page http://tookan.gforge.inria.fr/.

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. Tookan 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. Tookan has been used to find at least a dozen previously unknown flaws in commercially available devices.

The first results using Tookan were published in 2010 [47] and a six-month licence was granted to Boeing to use the tool. In 2011, a contract was signed with a major UK bank. Tookan is now the subject of a CSATT transfer action resulting in the hiring of an engineer, Romain Bardou, who started on September 1st, 2011. During 2012 Bardou and Steel implemented a new version of Tookan that is intended to form the technological basis for a spin-off company to be created in 2013.

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 is under review, and the software is being prepared for imminent release in January 2013.

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 in the Symbolic Model

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 and on an extension of ProVerif to prove more observational equivalences.

6.1.1. Verification of Protocols with Lists

Participants: Bruno Blanchet [correspondant], Miriam Paiola.

security protocols, symbolic model, automatic verification, Horn clauses, secrecy

We have designed a novel, simple technique for proving secrecy properties for security protocols that manipulate lists of unbounded length, for an unbounded number of sessions [32]. More specifically, our technique relies on the Horn clause approach used in the automatic verifier ProVerif: we show that if a protocol is proven secure by our technique with lists of length one, then it is secure for lists of unbounded length. Interestingly, this theorem relies on approximations made by our verification technique: in general, secrecy for lists of length one does not imply secrecy for lists of unbounded length. Our result can be used in particular to prove secrecy properties for group protocols with an unbounded number of participants and for some XML protocols (web services) with ProVerif.

6.1.2. Proving More Process Equivalences with ProVerif

Participants: Bruno Blanchet [correspondant], Vincent Cheval.

security protocols, symbolic model, automatic verification, observational equivalence, privacy

We have extended the automatic protocol verifier ProVerif in order to prove more observational equivalences [28]. ProVerif can prove observational equivalence between processes that have the same structure but differ by the messages they contain. In order to extend the class of equivalences that ProVerif handles, we extend the language of terms by defining more functions (destructors) by rewrite rules. In particular, we allow rewrite rules with inequalities as side-conditions, so that we can express tests "if then else" inside terms. Finally, we provide an automatic procedure that translates a process into an equivalent process that performs as many actions as possible inside terms, to allow ProVerif to prove the desired equivalence. These extensions have been implemented in ProVerif and allow us to automatically prove anonymity in the private authentication protocol by Abadi and Fournet.

6.2. Verification of Security Protocols in the Computational Model

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 worked on a compiler from CryptoVerif speficications to OCaml, and we have used CryptoVerif to verify the password-based protocol One-Encryption Key Exchange (OEKE).

6.2.1. Generation of Implementations Proved Secure in the Computational model

Participants: Bruno Blanchet [correspondant], David Cadé.

security protocols, computational model, implementation, verification, compiler

We have designed a novel approach for proving specifications of security protocols in the computational model and generating runnable implementations from such proved specifications. We rely on the computationally-sound protocol verifier CRYPTOVERIF for proving the specification, and we have implemented a compiler that translates a CRYPTOVERIF specification into an implementation in OCaml [26]. We have also proved that this compiler preserves security [27]. We have applied this compiler to the SSH Transport Layer protocol: we proved the authentication of the server and the secrecy of the session keys in this protocol and verified that the generated implementation successfully interacts with OpenSSH. The secrecy of messages sent over the SSH tunnel cannot be proved due to known weaknesses in SSH with CBC-mode encryption.

6.2.2. Proof of One-Encryption Key Exchange using CryptoVerif

Participant: Bruno Blanchet [correspondant].

security protocols, computational model, automatic proofs, formal methods, password-based authentication

We have obtained a mechanized proof of the password-based protocol One-Encryption Key Exchange (OEKE) using the computationally-sound protocol prover CRYPTOVERIF [25]. OEKE is a non-trivial protocol, and thus mechanizing its proof provides additional confidence that it is correct. This case study was also an opportunity to implement several important extensions of CRYPTOVERIF, useful for proving many other protocols. We have indeed extended CRYPTOVERIF to support the computational Diffie-Hellman assumption. We have also added support for proofs that rely on Shoup's lemma and additional game transformations. In particular, it is now possible to insert case distinctions manually and to merge cases that no longer need to be distinguished. Eventually, some improvements have been added on the computation of the probability bounds for attacks, providing better reductions. In particular, we improve over the standard computation of probabilities when Shoup's lemma is used, which allows us to improve the bound given in a previous manual proof of OEKE, and to show that the adversary can test at most one password per session of the protocol.

6.3. New Attacks on RSA PKCS#1 v1.5

Participants: Graham Steel [correspondant], Romain Bardou.

cryptographic hardware, security API, key management, vulnerabilities

RSA PKCS#1v1.5 is the most commonly used standard for public key encryption, used for example in TLS/SSL. It has been known to be vulnerable to a so-called padding-oracle attack since 1998 when Bleichenbacher described the vulnerability at CRYPTO. The attack, known as the "million message attack" was not thought to present a practical threat, due in part to the large number of oracle messages required. In a paper published at CRYPTO 2012 [22] we gave original modifications showing how the attack can be completed in a median of just 15 000 messages. The results lead to widespread interest, indicated by over 1400 downloads of the long version of the paper from the HAL webpage and articles in the New York Times, Boston Globe and Süddeutscher Zeitung.

6.4. Security Proofs for Revocation

Participants: Graham Steel [correspondant], Véronique Cortier, Cyrille Wiedling.

security API, key management, formal methods, security proofs

Revocation of expired or corrupted keys is a common feature of industrially deployed key management systems but an aspect that is almost always missing from formal models. We succeeded in adding revocation to a formal specification of a key management API allowing the proof of strong security properties after corrupted keys are revoked. In particular we showed a self-healing property whereby after a corrupted key expires, after a certain amount of time, the system is safe again. The work was published at ACM CCS 2012.

6.5. Discovering Concrete Attacks on Web Applications by Formal Analysis

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

web application security, formal methods, automated verification, vulnerabilities Social sign-on and social sharing are becoming an ever more popular feature of web applications. This success is largely due to the APIs and support 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. A formal analysis of these protocols must account for malicious websites and common web application vulnerabilities, such as cross-site request forgery and open redirectors. We model several configurations of the OAuth 2.0 protocol in the applied pi-calculus and verify them using ProVerif. Our models rely on WebSpi, a new library for modeling web applications and web-based attackers that is designed to help discover concrete website attacks. Our approach is validated by finding dozens of previously unknown vulnerabilities in popular websites such as Yahoo and WordPress, when they connect to social networks such as Twitter and Facebook. This work was published in CSF'12 [21].

To protect sensitive user data against server-side attacks, a number of security-conscious web applications have turned to client-side encryption, where only encrypted user data is ever stored in the cloud. We formally investigate the security of a number of such applications, including password managers, cloud storage providers, an e-voting website and a conference management system. We show that their security relies on both their use of cryptography and the way it combines with common web security mechanisms as implemented in the browser. We model these applications using the WebSpi web security library for ProVerif, we discuss novel attacks found by automated formal analysis, and we propose robust countermeasures. Some of the attacks we discovered were presented at WOOT'12 [24]. Our formal models and verified countermeasures are going to be presented at POST'13 [20].

6.6. Attacks and Proofs for TLS Implementations

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

cryptographic protocol, formal methods, automated verification, traffic analysis, vulnerabilities

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 list two new results below, both are submitted for review.

Websites commonly use HTTPS to protect their users' private data from network-based attackers. By combining public social network profiles with TLS traffic analysis, we present a new attack that reveals the precise identities of users accessing major websites. As a countermeasure, we propose a novel length-hiding scheme that leverages standard TLS padding to enforce website-specific privacy policies. We present several implementations of this scheme, notably a patch for GnuTLS that offers a rich length-hiding API and an Apache module that uses this API to enforce an anonymity policy for sensitive user files. Our implementations are the first to fully exercise the length-hiding features of TLS and our work uncovers hidden timing assumptions in recent formal proofs of these features. Compared to previous work, we offer the first countermeasure that is standards-based, provably secure, and experimentally effective, yet pragmatic, offering websites a precise trade-off between user privacy and bandwidth efficiency. This work is available as an Inria technical report [36].

We develop a verified reference implementation of TLS 1.2. Our code fully supports its wire formats, ciphersuites, sessions and connections, re-handshakes and resumptions, alerts and errors, and data fragmentation, as prescribed in the RFCs; it interoperates with mainstream web browsers and servers. At the same time, our code is carefully structured to enable its modular, automated verification, from its main API down to computational assumptions on its cryptographic algorithms. 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 is under review and will be released as an Inria technical report in January 2013.

7. Bilateral Contracts and Grants with Industry

7.1. Technology Transfer Grant

Inria CSATT Technology Transfer Action for Tookan. Following successful technology transfer projects around the Tookan software with Boeing and Barclays Bank, Inria have provided 12 months of funding for a software engineer (Romain Bardou) and 10 kEuros.

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.)

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.

8.2. 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

- We work closely with Microsoft Research in Cambridge, Redmond, and Bangalore (C. Fournet, N. Swamy, P. Naldurg)
- We work closely with University of Venice, Italy (R. Foccardi).

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Michael May (Faculty Lecturer, Kinneret College on the Sea of Galilee, Israel) visited us for three months as professeur invité.
- Sergio Maffeis (Imperial College, London) visited us as part of an ongoing collaboration.

8.4.1.1. Internships

- Jean Karim Zinzindohoue did his M1 stage with Karthikeyan Bhargavan. He won the "Prix du stage de recherche dit prix d'option" for his work on "Tracking Cryptographically Masked Flows in Android Applications"
- Antoine Delignat-Lavaud did his M2 stage with Karthikeyan Bhargavan on "Security Types for Web Applications"
- Chetan Bansal did a Master's stage with Karthikeyan Bhargavan on "Analysis and Verification of Security for Web Applications"
- Avinash Thummala did a Master's stage with Karthikeyan Bhargavan on "Verifying JavaCard Applets"
- Sneha Popley did a PhD summer internship with Karthikeyan Bhargavan on "Verifying Cryptographic Applications in Java"

8.4.2. Visits to International Teams

- Visits to Imperial College, London: Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Chetan Bansal
- Visits to Microsoft Research, Cambridge: Karthikeyan Bhargavan, Alfredo Pironti
- Visits to University of Birmingham: Ben Smyth, Miriam Paiola

9. Dissemination

9.1. Editorial Boards

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

9.2. Organizers

- Dagstuhl Analysis of Security APIs, November: Graham Steel
- ASA-6, Satellite of CSF, June: Graham Steel

9.3. Program Committees

- FCC – June 2012, Cambridge, MA, USA: Bruno Blanchet
- FM – August 2012, Paris, France: Bruno Blanchet
- CCS – October 2012, Ralleigh, NC, USA: Bruno Blanchet
- POST – March 2013, Rome, Italy: Bruno Blanchet
- PROOFS workshop - September 2012, September 2012: Graham Steel

- ACM SAC - Security Track 2012: Graham Steel

9.4. Vulnerability Reports

- R. Bardou and G. Steel (and their co-authors) found a series of vulnerabilities in cryptographic devices, resulting in a research paper and significant press coverage.
- K. Bhargavan and C. Bansal reported single sign-on vulnerabilities in Facebook, Yahoo, Helios and ConfiChair. These vulnerabilities were fixed on their recommendations, and the Prosecco team name was given credit on their websites.
- K. Bhargavan and A. Delignat-Lavaud reported encryption-related vulnerabilities in several commercial software packages, including 1Password, Roboform, LastPass, SpiderOak, and Wuala. These vulnerabilites were fixed on their recommendations and the Prosecco team was given credit on their websites.
- K. Bhargavan reported a CSP-related security vulnerability in Firefox, resulting in a security update.
- A. Delignat Lavaud reported a iframes-related security vulnerability in Firefox, resulting in a security update.

9.5. Teaching

- Karthikeyan Bhargavan taught TDs in computer science at Ecole Polytechnique, France
- Graham Steel taught invited Master's lectures at University of Venice, Italy
- Bruno Blanchet taught Master's lectures at the MPRI, France

9.6. Ph.D in progress

- David Cadé
  Computationally Proved Implementations of Security Protocols,
  since September 2009, 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
- Gavin Keighren, *A Type System for Security APIs*, since 2007 (to submit March 2013), advisors Graham Steel and David Aspinall (University of Edinburgh). Graham is now at EPI Prosecco.

9.7. Ph.D/Habilitation Committees

- Vincent Cheval – Ph.D. – 3 Dec. 2012 – ENS Cachan
  Automatic verification of cryptographic protocols: privacy-type properties
  Bruno Blanchet

9.8. Invited Talks

- Workshop on Computed-Aided Security, Grenoble, France, January: Bruno Blanchet
- ETAPS unifying invited speaker, Tallinn, Estonia March: Bruno Blanchet
- Workshop on Formal and Computational Cryptographic Proofs, Cambridge, UK (April): Bruno Blanchet
- Alan Turing Year workshop: Is Cryptographic Theory Practically Relevant?, Cambridge, 31 January - 2 February: Graham Steel
- PROOFS workshop, satellite of CHES, September: Graham Steel

- DIGITEO Forum, Ecole polytechnqiue November: Graham Steel
- Ruhr University Bochum Seminar, June: Graham Steel

9.9. Participation to Workshops and Conferences

- Workshop on Computed-Aided Security – January 2012, Grenoble, France: Bruno Blanchet, David Cadé, Miriam Paiola
- CryptoForma – March 2012, UK: Alfredo Pironti
- ETAPS – March 2012, Tallinn, Estonia]: Bruno Blanchet, David Cadé, Miriam Paiola
- Workshop on Formal and Computational Cryptographic Proofs – April 2012, Cambridge, UK: Bruno Blanchet
- IEEE S&P – May 2012, San Francisco, USA: Graham Steel
- CSF, FCC, ASA – June 2012, Cambridge MA, USA: Bruno Blanchet, David Cadé, Miriam Paiola, Graham Steel, Robert Künnemann, Karthikeyan Bhargavan, Romain Bardou
- Usenix Security – August 2012, Bellevue WA, USA: Karthikeyan Bhargavan, Antoine Delignat-Lavaud
- ARES – August 2012, Prague, Czech Republic: David Cadé
- CRYPTO – August 2012, Santa Barbara, USA: Graham Steel, Romain Bardou
- PROOFS – September 2012, Leuven Belgium: Graham Steel, Evmorfia-Iro Bartzia, Miriam Paiola
- MSR-Inria Workshop – November 2012, Cambridge UK: Karthikeyan Bhargavan, Alfredo Pironti

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, p. 102–146.

[2] C. BANSAL, K. BHARGAVAN, S. MAFFEIS. *Discovering Concrete Attacks on Website Authorization by Formal Analysis*, in "25th IEEE Computer Security Foundations Symposium (CSF'12)", Cambridge, MA, USA, IEEE, June 2012, p. 247–262.

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

[4] K. BHARGAVAN, C. FOURNET, R. CORIN, E. ZALINESCU. *Cryptographically verified implementations for TLS*, in "ACM Conference on Computer and Communications Security", 2008, p. 459-468.

[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, p. 445-456.

[6] B. BLANCHET. *Automatically Verified Mechanized Proof of One-Encryption Key Exchange*, in "25th IEEE Computer Security Foundations Symposium (CSF'12)", Cambridge, MA, USA, IEEE, June 2012, p. 325–339.

[7] B. BLANCHET, A. CHAUDHURI. *Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage*, in "IEEE Symposium on Security and Privacy", Oakland, CA, IEEE, May 2008, p. 417–431.

[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, p. 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, p. 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, p. 918-928.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] K. BHARGAVAN. *Towards the Automated Verification of Cryptographic Protocol Implementations*, École Normale Supérieure, May 2012, Mémoire d'habilitation.

Articles in International Peer-Reviewed Journals

[12] M. AVALLE, A. PIRONTI, R. SISTO. *Formal verification of security protocol implementations: a survey*, in "Formal Aspects of Computing", 2012, p. 1-25.

[13] 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, p. 3:1–3:32.

[14] V. CORTIER, B. SMYTH. *Attacking and fixing Helios: An analysis of ballot secrecy*, in "Journal of Computer Security", 2012, Accepted.

[15] A. PIRONTI, D. POZZA, R. SISTO. *Formally-Based Semi-Automatic Implementation of an Open Security Protocol*, in "Journal of Systems and Software", 2012, vol. 85, p. 835–849.

[16] A. PIRONTI, R. SISTO. *Safe Abstractions of Data Encodings in Formal Security Protocol Models*, in "Formal Aspects of Computing", 2012, p. 1-43.

Invited Conferences

[17] B. BLANCHET. *Security Protocol Verification: Symbolic and Computational Models*, in "First Conference on Principles of Security and Trust (POST'12)", Tallinn, Estonia, P. DEGANO, J. GUTTMAN (editors), Lecture Notes on Computer Science, Springer Verlag, March 2012, vol. 7215, p. 3–29.

International Conferences with Proceedings

[18] G. BANA, P. ADÃO, H. SAKURADA. *Computationally Complete Symbolic Attacker in Action*, in "IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)", 2012, p. 546-560.

[19] G. BANA, H. COMON-LUNDH. *Towards Unconditional Soundness: Computationally Complete Symbolic Attacker*, in "2nd Conference on Principles of Security and Trust (POST 2013)", 2012, p. 189-208.

[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), Lecture Notes on Computer Science, Springer Verlag, March 2013, To appear.

[21] C. BANSAL, K. BHARGAVAN, S. MAFFEIS. *Discovering Concrete Attacks on Website Authorization by Formal Analysis*, in "25th IEEE Computer Security Foundations Symposium (CSF'12)", Cambridge, MA, USA, IEEE, June 2012, p. 247–262.

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

[23] P. BETTASSA COPET, A. PIRONTI, D. POZZA, R. SISTO, P. VIVOLI. *Visual Model-Driven Design, Verification and Implementation of Security Protocols*, in "IEEE International Symposium on High Assurance Systems Engineering (HASE 12)", IEEE Computer Security, 2012, Short paper.

[24] K. BHARGAVAN, A. DELIGNAT-LAVAUD. *Web-based Attacks on Host-Proof Encrypted Storage*, in "6th USENIX Workshop on Offensive Technologies (WOOT'12)", Usenix, aug 2012.

[25] B. BLANCHET. *Automatically Verified Mechanized Proof of One-Encryption Key Exchange*, in "25th IEEE Computer Security Foundations Symposium (CSF'12)", Cambridge, MA, USA, IEEE, June 2012, p. 325–339.

[26] D. CADÉ, B. BLANCHET. *From Computationally-proved Protocol Specifications to Implementations*, in "7th International Conference on Availability, Reliability and Security (AReS 2012)", Prague, Czech Republic, IEEE, August 2012, p. 65–74.

[27] 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), Lecture Notes on Computer Science, Springer Verlag, March 2013, to appear.

[28] 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), Lecture Notes on Computer Science, Springer Verlag, March 2013, to appear.

[29] 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, p. 918-928.

[30] R. KÜNNEMANN, G. STEEL. *YubiSecure? Formal Security Analysis Results for the Yubikey and YubiHSM*, in "Preliminary Proceedings of the 8th Workshop on Security and Trust Management (STM'12)", Pisa, Italy, A. JØSANG, P. SAMARATI (editors), September 2012.

[31] M. J. MAY, K. BHARGAVAN. *Towards Unified Authorization for Android*, in "5th International Symposium on Engineering Secure Software and Systems (ESSoS 2013)", 2013, To appear.

[32] M. PAIOLA, B. BLANCHET. *Verification of Security Protocols with Lists: from Length One to Unbounded Length*, in "First Conference on Principles of Security and Trust (POST'12)", Tallinn, Estonia, P. DEGANO, J. GUTTMAN (editors), Lecture Notes on Computer Science, Springer Verlag, March 2012, vol. 7215, p. 69–88.

Scientific Books (or Scientific Book chapters)

[33] B. BLANCHET. *Mechanizing Game-Based Proofs of Security Protocols*, in "Software Safety and Security - Tools for Analysis and Verification", T. NIPKOW, O. GRUMBERG, B. HAUPTMANN (editors), NATO Science for Peace and Security Series – D: Information and Communication Security, IOS Press, May 2012, vol. 33, p. 1–25, Proceedings of the summer school MOD 2011.

Research Reports

[34] R. BARDOU, R. FOCARDI, Y. KAWAMOTO, L. SIMIONATO, G. STEEL, J.-K. TSAY. *Efficient Padding Oracle Attacks on Cryptographic Hardware*, Inria, April 2012, n° RR-7944, 19, http://hal.inria.fr/hal-00691958.

[35] V. CORTIER, G. STEEL, C. WIEDLING. *Revoke and Let Live: A Secure Key Revocation API for Cryptographic Devices*, Inria, July 2012, n° RR-7949, 41, http://hal.inria.fr/hal-00721945.

[36] A. PIRONTI, P.-Y. STRUB, K. BHARGAVAN. *Identifying Website Users by TLS Traffic Analysis: New Attacks and Effective Countermeasures*, Inria, September 2012, n° RR-8067, http://hal.inria.fr/hal-00732449.

[37] B. SMYTH, M. D. RYAN, L. CHEN. *Formal analysis of privacy in Direct Anonymous Attestation schemes*, Cryptology ePrint Archive, 2012, n° 2012/650.

References in notes

[38] M. ABADI, B. BLANCHET. *Analyzing Security Protocols with Secrecy Types and Logic Programs*, in "Journal of the ACM", January 2005, vol. 52, n° 1, p. 102–146.

[39] M. ABADI, B. BLANCHET, C. FOURNET. *Just Fast Keying in the Pi Calculus*, in "ACM Transactions on Information and System Security (TISSEC)", July 2007, vol. 10, n° 3, p. 1–59.

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

[41] 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, p. 445–456.

[42] 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, p. 123–135.

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

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

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

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

[47] 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, p. 260-269.

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

[49] 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, p. 1211-1245.

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

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

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

[53] 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, p. 266-278.