Activity Report 2011

Project-Team INDES

Secure Diffuse Programming

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Distributed Systems and Services

Table of contents

Project-Team INDES

Keywords: Programming Languages, Compiling, Security, Concurrency, Web

1. Members

Research Scientists

Gérard Berry [Research Director, Inria, HdR]
Gérard Boudol [Research Director, Inria, HdR]
Ilaria Castellani [Research Scientist, Inria]
Tamara Rezk [Research Scientist, Inria]
Bernard Serpette [Research Scientist, Inria]
Manuel Serrano [Team Leader, Research Director, Inria, HdR]
Frédéric Boussinot [Research Director, CMA, HdR]

Faculty Member

Christian Queinnec [Professor, University Pierre et Marie Curie - Paris 6, HdR]

PhD Students

Zhengqin Luo [MENRT]
Cyprien Nicolas [Inria]
Johan Grande [Inria]
Pejman Attar [Inria]
José Santos [Inria]

Post-Doctoral Fellow

Thomas Gazagnaire [Inria]

Administrative Assistant

Nathalie Bellesso [Inria]

2. Overall Objectives

2.1. Overall Objectives

The goal of the Indes team is to study models for diffuse computing and develop languages for secure diffuse applications. Diffuse applications, of which Web 2.0 applications are a notable example, are the new applications emerging from the convergence of broad network accessibility, rich personal digital environment, and vast sources of information. Strong security guarantees are required for these applications, which intrinsically rely on sharing private information over networks of mutually distrustful nodes connected by unreliable media.

Diffuse computing requires an original combination of nearly all previous computing paradigms, ranging from classical sequential computing to parallel and concurrent computing in both their synchronous / reactive and asynchronous variants. It also benefits from the recent advances in mobile computing, since devices involved in diffuse applications are often mobile or portable.

The Indes team contributes to the whole chain of research on models and languages for diffuse computing, going from the study of foundational models and formal semantics to the design and implementation of new languages to be put to work on concrete applications. Emphasis is placed on correct-by-construction mechanisms to guarantee correct, efficient and secure implementation of high-level programs. The research is partly inspired by and built around Hop, the web programming model proposed by the former Mimosa team, which takes the web as its execution platform and targets interactive and multimedia applications.

3. Scientific Foundations

3.1. Parallelism, concurrency, and distribution

Concurrency magagement is at the heart of diffuse programming. Since the execution platforms are highly heterogeneous, many different concurrency principles and models may involved. Asynchronous concurrency is the basis of shared-memory process handling within multiprocessor or multicore computers, of direct or fifo-based message passing in distributed networks, and of fifo- or interrupt-based event handling in web-based human-machine interaction or sensor handling. Synchronous or quasi-synchronous concurrency is the basis of signal processing, of real-time control, and of safety-critical information acquisition and display. Interfacing existing devices based on these different concurrency principles within HOP or other diffuse programming languages will require better understanding of the underlying concurrency models and of the way they can nicely cooperate, a currently ill-resolved problem.

3.2. Web and functional programming

We are studying new paradigms for programming Web applications that rely on multi-tier functional programming [4]. We have have created a Web programming environment named HOP. It relies on a single formalism for programming the server-side and the client-side of the applications as well as for configuring the execution engine.

HOP is a functional language based on the SCHEME programming language. That is, it is a strict functional language, fully polymorphic, supporting side effects, and dynamically type-checked. HOP is implemented as an extension of the BIGLOO compiler that we develop [5]. In the past, we have extensively studied static analyses (type systems and inference, abstract interpretations, as well as classical compiler optimizations) to improve the efficiency of compilation in both space and time.

3.3. Security of diffuse programs

The main goal of our security research is to provide scalable and rigorous language-based techniques that can be integrated into multi-tier compilers to enforce the security of diffuse programs. Research on language-based security has been carried on before in former Inria teams [2], [1]. In particular previous research has focused on controlling information flow to ensure confidentiality.

Typical language-based solutions to these problems are founded on static analysis, logics, provable cryptography, and compilers that generate correct code by construction [3]. Relying on the multi-tier programming language HOP that tames the complexity of writing and analysing secure diffuse applications, we are studying language-based solutions to prominent web security problems such as code injection and cross-site scripting, to name a few.

4. Application Domains

4.1. Web programming

Along with games, multimedia applications, electronic commerce, and email, the web has popularized computers in everybody's life. The revolution is engaged and we may be at the dawn of a new era of computing where the web is a central element. The web constitutes an infrastructure more versatile, polymorphic, and open, in other words, more powerful, than any dedicated network previously invented. For this very reason, it is likely than most of the computer programs we will write in the future, for professional purposes as well as for our own needs, will extensively rely on the web.

In addition to allowing reactive and graphically pleasing interfaces, web applications are de facto distributed. Implementing an application with a web interface makes it instantly open to the world and accessible from much more than one computer. The web also partially solves the problem of platform compatibility because it physically separates the rendering engine from the computation engine. Therefore, the client does not have to make assumptions on the server hardware configuration, and vice versa. Lastly, HTML is highly durable. While traditional graphical toolkits evolve continuously, making existing interfaces obsolete and breaking backward compatibility, modern web browsers that render on the edge web pages are still able to correctly display the web pages of the early 1990's.

For these reasons, the web is arguably ready to escape the beaten track of n-tier applications, CGI scripting and interaction based on HTML forms. However, we think that it still lacks programming abstractions that minimize the overwhelming amount of technologies that need to be mastered when web programming is involved. Our experience on reactive and functional programming is used for bridging this gap.

4.2. Multimedia

Electronic equipments are less and less expensive and more and more widely spread out. Nowadays, in industrial countries, computers are almost as popular as TV sets. Today, almost everybody owns a mobile phone. Many are equipped with a GPS or a PDA. Modem, routers, NASes and other network appliances are also commonly used, although they are sometimes sealed under proprietary packaging such as the Livebox or the Freebox. Most of us evolve in an electronic environment which is rich but which is also populated with mostly isolated devices.

The first multimedia applications on the web have appeared with the Web 2.0. The most famous ones are Flickr, YouTube, or Deezer. All these applications rely on the same principle: they allow roaming users to access the various multimedia resources available all over the Internet via their web browser. The convergence between our new electronic environment and the multimedia facilities offered by the web will allow engineers to create new applications. However, since these applications are complex to implement this will not happen until appropriate languages and tools are available. In the Indes team, we develop compilers, systems, and libraries that address this problem.

4.3. House Automation

The web is the de facto standard of communication for heterogeneous devices. The number of devices able to access the web is permanently increasing. Nowadays, even our mobile phones can access the web. Tomorrow it could even be the turn of our wristwatches! The web hence constitutes a compelling architecture for developing applications relying on the "ambient" computing facilities. However, since current programming languages do not allow us to develop easily these applications, ambient computing is currently based on ad-hoc solutions. Programming ambient computing via the web is still to be explored. The tools developed in the Indes team allow us to build prototypes of a web-based home automation platform. For instance, we experiment with controlling heaters, air-conditioners, and electronic shutters with our mobile phones using web GUIs.

5. Software

5.1. Introduction

Most INDES software packages, even the older stable ones that are not described in the following sections are freely available on the Web. In particular, some are available directly from the INRIA Web site:

http://www.inria.fr/centre/sophia/innovation

Most other software packages can be downloaded from the INDES Web site:

http://www-sop.inria.fr/teams/indes

5.2. Functional programming

Participants: Frédéric Boussinot [Inria], Thomas Gazagnaire [Inria], Zhengqin Luo [Inria], Cyprien Nicolas [Inria], Tamara Rezk [Inria], Bernard Serpette [Inria], Manuel Serrano [correspondant].

5.2.1. The Bigloo compiler

The programming environment for the Bigloo compiler [5] is available on the INRIA Web site at the following URL: http://www-sop.inria.fr/teams/indes/fp/Bigloo. The distribution contains an optimizing compiler that delivers native code, JVM bytecode, and .NET CLR bytecode. It contains a debugger, a profiler, and various Bigloo development tools. The distribution also contains several user libraries that enable the implementation of realistic applications.

BIGLOO was initially designed for implementing compact stand-alone applications under Unix. Nowadays, it runs harmoniously under Linux and MacOSX. The effort initiated in 2002 for porting it to Microsoft Windows is pursued by external contributors. In addition to the native back-ends, the BIGLOO JVM back-end has enabled a new set of applications: Web services, Web browser plug-ins, cross platform development, etc. The new BIGLOO .NET CLR back-end that is fully operational since release 2.6e enables a smooth integration of Bigloo programs under the Microsoft .NET environment.

5.2.2. Camloo

Camloo is a caml-light to bigloo compiler, which was developed few years ago to target bigloo 1.6c. New major releases 0.4.x of camloo have been done to support bigloo 3.4 and bigloo 3.5. Camloo make it possible for the user to develop seamlessly a multi-language project, where some files are written in caml-light, in C, and in bigloo. Unlike the previous versions of camloo, 0.4.x versions do not need a modified bigloo compiler to obtain good performance. Currently, the only supported backend for camloo is bigloo/C. We are currently rewriting the runtime of camloo in bigloo to get more portability and to be able to use HOP and camloo together.

5.2.3. The FunLoft language

FunLoft (described in http://www-sop.inria.fr/teams/indes/rp/FunLoft) is a programming language in which the focus is put on safety and multicore.

FunLoft is built on the model of FairThreads which makes concurrent programming simpler than usual preemptive-based techniques by providing a framework with a clear and sound semantics. FunLoft is designed with the following objectives:

- provide a safe language, in which, for example, data-races are impossible.
- control the use of resources (CPU and memory), for example, memory leaks cannot occur in FunLoft programs, which always react in finite time.
- have an efficient implementation which can deal with large numbers of concurrent components.
- benefit from the real parallelism offered by multicore machines.

A first experimental version of the compiler is available on the Reactive Programming site http://www-sop.inria.fr/teams/indes/rp. Several benchmarks are given, including cellular automata and simulation of colliding particles.

5.3. Web programming

Participants: Gérard Berry [Inria], Cyprien Nicolas [Inria], Manuel Serrano [correspondant].

5.3.1. The HOP web programming environment

HOP is a higher-order language designed for programming interactive web applications such as web agendas, web galleries, music players, etc. It exposes a programming model based on two computation levels. The first one is in charge of executing the logic of an application while the second one is in charge of executing the graphical user interface. HOP separates the logic and the graphical user interface but it packages them together and it supports strong collaboration between the two engines. The two execution flows communicate through function calls and event loops. Both ends can initiate communications.

The HOP programming environment consists in a web *broker* that intuitively combines in a single architecture a web server and a web proxy. The broker embeds a HOP interpreter for executing server-side code and a HOP client-side compiler for generating the code that will get executed by the client.

An important effort is devoted to providing HOP with a realistic and efficient implementation. The HOP implementation is *validated* against web applications that are used on a daily-basis. In particular, we have developed HOP applications for authoring and projecting slides, editing calendars, reading RSS streams, or managing blogs.

HOP has won the software *open source contest* organized by the ACM Multimedia Conference 2007 http://mmc36.informatik.uni-augsburg.de/acmmm2007/. It is released under the GPL license. It is available at http://hop.inria.fr

5.4. Language-based security

Participants: Zhengqin Luo [Inria], Tamara Rezk [correspondant].

5.4.1. CFlow

The prototype compiler "CFlow" takes as input code annotated with information flow security labels for integrity and confidentiality and compiles to F# code that implements cryptography and protocols that satisfy the given security specification.

Cflow has been coded in F#, developed mainly on Linux using mono (as a substitute to .NET), and partially tested under Windows (relying on .NET and Cygwin). The code is distributed under the terms of the CeCILL-B license http://www.msr-inria.inria.fr/projects/sec/cflow/index.html.

5.4.2. FHE type-checker

We have developed a type checker for programs that feature modern cryptographic primitives such as fully homomorphic encryption. The type checker is thought as an extension of the "CFlow" compiler developed last year on the same project. It is implemented in F#. The code is distributed under the terms of the CeCILL-B license http://www.msr-inria.inria.fr/projects/sec/cflow/index.html.

5.4.3. Mashic compiler

The Mashic compiler is applied to mashups with untrusted scripts. The compiler generates mashups with sandboxed scripts, secured by the same origin policy of the browsers. The compiler is written in Bigloo and can be found at http://www.mashic.net.

5.5. Old software

5.5.1. Skribe

SKRIBE is a functional programming language designed for authoring documents, such as Web pages or technical reports. It is built on top of the SCHEME programming language. Its concrete syntax is simple and looks familiar to anyone used to markup languages. Authoring a document with SKRIBE is as simple as with HTML or LaTeX. It is even possible to use it without noticing that it is a programming language because of the conciseness of its original syntax: the ratio *tag/text* is smaller than with the other markup systems we have tested.

Executing a SKRIBE program with a SKRIBE evaluator produces a target document. It can be HTML files for Web browsers, a LaTeX file for high-quality printed documents, or a set of *info* pages for on-line documentation.

5.5.2. Scheme2JS

Scm2JS is a Scheme to JavaScript compiler distributed under the GPL license. Even though much effort has been spent on being as close as possible to R5RS, we concentrated mainly on efficiency and interoperability. Usually Scm2JS produces JavaScript code that is comparable (in speed) to hand-written code. In order to achieve this performance, Scm2JS is not completely R5RS compliant. In particular it lacks exact numbers.

Interoperability with existing JavaScript code is ensured by a JavaScript-like dot-notation to access JavaScript objects and by a flexible symbol-resolution implementation.

Scm2JS is used on a daily basis within HOP, where it generates the code which is sent to the clients (web-browsers). Scm2JS can be found at http://www-sop.inria.fr/indes/scheme2js.

6. New Results

6.1. Security

Participants: Ilaria Castellani, Zhengqin Luo, Tamara Rezk [correspondant], José Santos, Manuel Serrano.

6.1.1. Secure session calculi

We have pursued our work on controlling information flow in session calculi, started in previous years in collaboration with colleagues from the university of Torino. We also started investigating a notion of (objective) reputation for principals participating in sessions. The reputation of a principal is based on her previous behaviour as a user of a service. A principal's reputation can be checked both by the service itself, before admitting again the principal as a user, or by other principals to evaluate the reputation of the current users before they join a service (we consider multi-user services). We plan to apply this idea to refine our previous work on information flow control in multiparty sessions, by considering reputations built on the "security behaviour" of principals.

In the work "Information flow safety in multiparty sessions" [11], we consider a calculus for multiparty sessions enriched with security levels for messages. We propose a monitored semantics for this calculus, which blocks the execution of processes as soon as they attempt to leak information. We illustrate the use of our monitored semantics with various examples, and show that the induced safety property implies the security property studied previously for the same calculus. This work was presented at the 18th International Workshop on Expressiveness in Concurrency (EXPRESS'11).

In the work "A Reputation System for Multirole Sessions" [10], we extend role-based multiparty sessions with reputations and policies associated with principals. The reputation associated with a principal in a service is built by collecting her relevant behaviour as a participant in sessions of the service. The service checks the reputation of principals before allowing them to take part in a session, and decides whether to accept them or not depending on their reputation and on the role they want to play. Furthermore, principals can declare policies that must be fulfilled by the other participants of the same service. These policies are used by principals to check the reputation of the current participants and to decide accordingly whether or not to join the service. Our approach is illustrated by an example describing a real-world protocol. This work was presented at the 6th International Symposium on Trustworthy Global Computing (TGC'11).

Both [11] and [10] were partially funded by the ANR-08-EMER-010 grant PARTOUT.

6.1.2. Automatic Code Injection Prevention for Web Applications

We propose a new technique based on multitier compilation for preventing code injection in web applications. It consists in adding an extra stage to the client code generator which compares the dynamically generated code with the specification obtained from the syntax of the source program. No intervention from the programmer is needed. No plugin or modification of the web browser is required. The soundness and validity of the approach are proved formally by showing that the client compiler can be fully abstract. The practical interest of the approach is proved by showing the actual implementation in the HOP environment.

This work was presented in TOSCA'11 and appeared in the LNCS series [13]. See also software section.

6.1.3. A Certified Lightweight Non-Interference Java Bytecode Verifier

We propose a type system to verify the non-interference property in the Java Virtual Machine. We verify the system in the Coq theorem prover.

This work will appear in the journal of Mathematical Structures in Computer Science [8].

6.1.4. Information-flow types for homomorphic encryptions

We develop a flexible information-flow type system for a range of encryption primitives, precisely reflecting their diverse functional and security features. Our rules enable encryption, blinding, homomorphic computation, and decryption, with selective key re-use for different types of payloads. We show that, under standard cryptographic assumptions, any well-typed probabilistic program using encryptions is secure (that is, computationally non-interferent) against active adversaries, both for confidentiality and integrity. We illustrate our approach using ElGamal and Paillier encryption.

We present two applications of cryptographic verification by typing: (1) private search on data streams; and (2) the bootstrapping part of Gentry's fully homomorphic encryption.

We provide a prototype typechecker for our system.

This work appeared in CCS'11 [12]. See also software section.

6.1.5. The Mashic compiler

Mashups are a prevailing kind of web applications integrating external gadget APIs often written in the Javascript programming language. Writing secure mashups is a challenging task due to the heterogeneity of existing gadget APIs, the privileges granted to gadgets during mashup executions, and Javascript's highly dynamic environment.

We propose a new compiler, called Mashic, for the automatic generation of secure Javascript-based mashups from existing mashup code. The Mashic compiler can effortlessly be applied to existing mashups based on a wide-range of gadget APIs. It offers security and correctness guarantees. Security is achieved by using the Same Origin Policy. Correctness is ensured in the presence of benign gadgets, that satisfy confidentiality and integrity constrains with regard to the integrator code. The compiler has been successfully applied to real world mashups based on Google maps, Bing maps, YouTube, and Zwibbler APIs.

See also software section.

6.1.6. Secure Information Flow by Self-Composition

Information flow policies are confidentiality policies that control information leakage through program execution. A common means to enforce secure information flow is through information flow type systems. Although type systems are compositional and usually enjoy decidable type checking or inference, their extensibility is very poor: type systems need to be redefined and proven sound for each new single variation of security policy and programming language for which secure information flow verification is desired. In contrast, program logics offer a general mechanism to enforce a variety of safety policies, and for this reason are favored in Proof Carrying Code, a promising security architecture for mobile code. However, the encoding of information flow policies in program logics is not straightforward, because they refer to a relation between two program executions. The purpose of this work is to investigate logical formulations of secure information

flow based on the idea of self-composition, that reduces the problem of secure information flow of a program P to a safety property for program P composed with itself.

This work appeared in the special issue of MSCS of PLID [7].

6.1.7. Secure Information flow enforcement techniques for dynamic security policies

We performed a comprehensive investigation of alternative static mechanisms to enforce information flow policies considering a setting in which programs run under an authority that is only known at runtime and that yields a relaxation of the base security policy. The devised method aims at eliminating the need to reanalyse a program each time its authority changes. The soundness of the proposed approach was established for a concurrent higher-order imperative lambda calculus with reference creation. This work resulted in the report "Typing Illegal Information Flows as Program Effects" available at http://www-sop.inria.fr/members/Jose.Santos/reportInfFlow.pdf.

6.2. Models, semantics, and languages

Participants: Pejman Attar, Gérard Berry [correspondant], Gérard Boudol, Frédéric Boussinot, Johan Grande, Cyprien Nicolas, Manuel Serrano.

6.2.1. HipHop

HipHop is a new Domain Specific Language for HOP dedicated to request and event orchestration. HipHop follows the synchronous reactive model of the Esterel and ReactiveC languages, originally developed for embedded systems programming. It is based on synchronous concurrency and preemption primitives, which are known to be key components for the modular design of complex temporal behaviors. Although the language is concurrent, the generated code is purely sequential and thread-free. HipHop is translated to Hop code to be interpreted by the Hop runtime, either on server or client sides. HipHop has been described in a paper [9] accepted at the new International Workshop on Programming Language And Systems Technologies for Internet Clients (Plastic 2011).

6.2.2. Tesard

Tesard is a programming language of the Caml family, designed to offer simple constructs for shared memory concurrency and a deadlock-free semantics.

It features in particular the 2 following constructs:

- `thread e` which launches a thread that will execute expression `e`, and returns immediately;
- `lock x in e` which takes a lock on mutable value `x` (possibly waiting to be able to do so), executes expression `e`, releases the lock on `x` and returns the result of the execution of `e`.

A type and effect system is used at compile-time to:

- associate a mutex with every mutable value
- make the `lock ... in...` construct implement deadlock avoidance.

The language is implemented as an interpreter and a bytecode compiler. It is a fork of Llama Light, which is itself derived from Caml. Llama Light is functionnaly roughly equivalent to Caml Light, but a large part of its code comes from OCaml, along with the threads library and runtime that we ported ourselves.

Several key parts of the language have been implemented: the runtime and most of the type and effect system, including inference of the creation of mutexes (`let region r in ...`) but not region polymorphism yet. The project is not in a usable state (no release has been made yet). Development versions are publicly available via GitHub at https://github.com/nahoj/llama.

6.2.3. Synchronous orchestration and beyond

We studied DSL, an orchestration language based on the synchronous/reactive model. In DSL, systems are composed of several sites executed asynchronously. Within each site, scripts are run in a synchronous parallel way. Scripts may call functions that are treated in an abstract way: their effect on the memory is not considered, but only their "orchestration", *i.e.,* the organisation of their calls in time and in place (the site where they are called). The mapping of sites onto cores allows one to benefit from multicore architectures. Two properties are required by DSL: reactivity of sites and absence of interferences between scripts run by distinct sites. We also introduced DSLM, which adds a memory level to DSL and a way to automatically adapt the execution to get a maximal use of the available cores. This work, presented respectively in [18] and [15], was funded by the ANR-08-EMER-010 grant PARTOUT.

6.3. Web programming

Participants: Zhengqin Luo, Cyprien Nicolas, Tamara Rezk, Bernard Serpette, Manuel Serrano [correspondant].

6.3.1. A new evaluator for Hop

At the time where Hop programs were basic scripts, the performance of the server-side interpreter was not a concern. An inefficient interpreter was acceptable. As Hop expanded, Hop programs got larger and more complex. A more efficient interpreter was necessary. Therefore, this year we have design and implemented a new interpreter for Hop. It is compact, its whole implementation counting no more than 2.5 KLOC. It is more than twice faster than the old interpreter and consumes less than a third of its memory. The architecture and the performance of the new interpreter are described in [14].

6.3.2. Abstraction of Hop with a bicolored lambda-calculus

We have studied an extension of the λ-calculus where each expression has a color. These colors designate the sites where expressions are evaluated, *i.e.*, in the server or in the client. Colors are similar to the `$` and `~` annotations of Hop. With this, we have defined a transformation, using β-expansion, which groups together expressions with the same color. Correction, confluence and termination of the transformation was verified using the Coq system and its description was described in paper to appear in 2012 [16].

Following Hop's syntax, `$` mentions that the followed expression is evaluated on the server while the `~` character introduce client code. Inside the client code it is allowed to reintroduce some server expressions by reusing a `$`. We can imagine, for example, that the action associated to a client's button is dependent on some server's data (order number, proxies, data bases, ...). This kind of example is depicted with the abstract syntax tree in the upper left part of figure 1.

From the server's point of view, the client's code following a `~` is ignored, but the environment in which this client code is activated must be preserved for all `$` inside this code. Therefore, there exists a *relation* between a node `~` and its including `$`. These relations are depicted by the dashed arrows in the upper right tree of Figure 1. The proposed transformation highlights this relation and is shown in the bottom of the figure where it can be observed that the transformation groups together expressions of the same color.

6.3.3. HopTeX, an Hop application for authoring documents

HopTeX is a Hop application for authoring Html and LaTeX documents. The content of the document is either expressed in Html or in a blending of Html and a dedicated wiki syntax, for the sake of conciseness and readability. The rendering of the document is expressed by a set of Css rules. The main originality of HopTeX is to consider LaTeX as a new media type for Html and to express the compilation from Html to LaTeX by the means of dedicated style sheet rules. HopTeX can then be used to generate high quality documents for both paper printed version and electronic version.

HopTeX is implemented in Hop, a multi-tier programming language for the Web 2.0. This implementation extensively relies on two facilities generally only available on the client-side that Hop also supports on the server-side of the application: DOM manipulations and Css server-side resolutions.

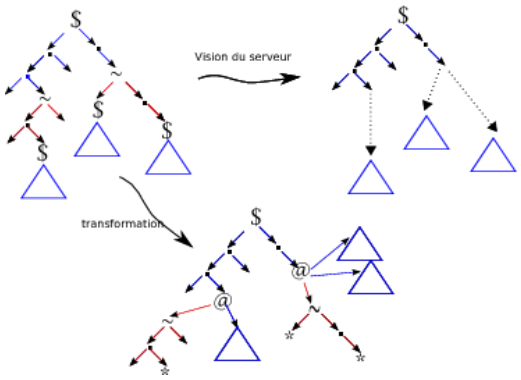

Figure 1. Example of Hop colored tree

The online version of the paper describing HOPTEX [17] is available at the HOPTEX web page (http://hop.inria.fr/hop/weblets/homepage?weblet=hoptex). Contrary to the PDF version published in the proceeding of the workshop, the online version is convenient for both desktop computers and Smartphones.

7. Contracts and Grants with Industry

7.1. Contracts and Grants with Industry

7.1.1. DGE SmartImmo

The SmartImmo DGE project of the world class ICT cluster SCS (Solutions Communicantes Sécurisées) started in 2009. It aimed at controlling and reducing the costs associated with building management. The duration of the project is 2 years, and the funding is 75k Euros. In the context of this project, the INDES team focuses on making the Hop Web development kit compatible with industrial house automation systems.

7.1.2. Collaboration with Xiring

In 2011, Tamara Rezk collaborated with a french company based on Paris, Xiring. She visited the company several times in 2011 to carry out this collaboration.

7.1.3. Microsoft Research and Inria Joint Lab

Since 2007, Tamara Rezk is part of the Secure Distributed Computations and their Proofs project of the MSR-Inria lab in Saclay. She travelled several times in 2011 to visit the lab and continue with several collaborations concerning the project.

8. Partnerships and Cooperations

8.1. National initiatives

8.1.1. ANR DEFIS ParTout

The PARTOUT project (PARTOUT = PARallélisme parTOUT) is funded by the ANR Défis programme for 4 years, starting January 2009. The partners of this project are the teams INDES (coordinator), CNAM/CÉDRIC, and LRI, Université d'Orsay.

8.1.2. ANR DEFIS PWD

The PWD project (for "Programmation du Web diffus") has been funded by the ANR Défis programme for 4 years, starting November 2009. The partners of this project are the teams INDES (coordinator), LIP6 at University Pierre et Marie Curie and PPS at University Denis Diderot.

8.1.3. MEALS

The MEALS project (Mobility between Europe and Argentina applying Logics to Systems), IRSES program, started October 1st (2011), and will end September 30th, 2015. The project goals cover three aspects of formal methods: specification (of both requirement properties and system behavior), verification, and synthesis. The Indes members are involved in the task of Security and Information Flow Properties (WP3). The partners in this task include University of Buenos Aires, University of Cordoba, INRIA (together with Catuscia Palamidessi, Kostas Chatzikokolakis, Miguel Andrés) and University of Twente.

8.1.4. CIRIC

Indes participated in the proposal of the CIRIC project, a joint lab between Inria and Chile, that will start in 2012. Indes members are involved in the line: Internet Research and Development.

9. Dissemination

9.1. Seminars and conferences

- **Pejman Attar** attended the conference MSR'11 in Lille, where he presented the work [15]. In November/December 2011, he participated in the SYNCHRON'11 5 day-workshop in Melun, where he presented [15] and the future perspectives of his work.
- **Gérard Berry** gave the following conferences:
  - *Challenges and Constructive Solutions for Complex Embedded Systems*, University of Bordeaux.
  - *Constructive Semantics for Discrete and Continuous Systems*, University of Barcelona.
  - Keynote Talk *hallenges and Constructive Solutions for Complex Embedded Systems* Embedded Systems Week, Taipei.
  - *Logic and Digital Circuits: from Practice to Theory* Workshop Analysing programs: logic to the rescue, 14th Congress of Logic, Methodology and Philosophy of Science, Nancy.
  - *Embedded Systems: Scientific Questions and Progress* European Commission study project on Embedded Systems Design, Bruxelles.
  - *Lecture « Hardware and software synchronous design and verification with Esterel v7* aux Microsoft Research Laboratories, Mountain View, CA.
- **Gérard Boudol** participated in a one-week Seminar in Dagstuhl on Multi-Core Memory Models and Concurrency Theory, where he gave a talk. He also gave a talk at the GG&JJ workshop in honour of Gérard Berry and Jean-Jacques Lévy, held in Gerardmer.
- **Ilaria Castellani** participated in the Behavioural Types workshop, in Lisbon, where she presented a preliminary version of the work [11].

  In May and October, she visited the university of Torino to pursue the collaborative work [11] and [10].

  In September, she attended the conference CONCUR 2010, as well as the associated workshop EXPRESS'11, where she presented the work [11], and the symposium on Trustworthy Global Computing (TGC'11), where the work [10] was presented by M. Dezani as an invited talk. The three events were held jointly in Aachen.

- **Zhengqin Luo** participated in the Cross-Seminar at INRIA in January, where he presented his work on semantics and security of web applications. In March, he participated to ETAPS conference, and presented his work on code injection prevention in TOSCA [13]. He visited Microsoft Research Cambridge from July to September, where he gave a talk on the work of automated mashup isolation in July. In October, he defended his thesis on the title of "Semantics and Security of Web Application" at INRIA [6].
- **Cyprien Nicolas** participated at the SPLASH conference organized by the ACM, where he gave three talks. The first talk was at the Scheme workshop annual conference about Manuel Serrano's work on HopTeX. The second was at DLS'11 about the new Hop interpreter, paper by Manuel Serrano and Bernard Serpette. Last talk was presenting Cyprien's work on HipHop (work done with Gérard Berry and Manuel Serrano) at the new International Workshop on Programming Language And Systems Technologies for Internet Clients (PLASTIC 2011). Cyprien also presented HipHop at Synchron'11 near Fontainebleau.
- **José Santos** gave a seminar entitled "Coping with dynamically allowed flow policies" concerning his work on information flow for the SQIG research group at IST Lisbon (in April 2011).
- **Tamara Rezk** presented her work in the FCC workshop, June 2011, Paris. In June 2011, she gave a seminar in LSV, where she was invited. In October 2011, she attended CCS'11 where her work was presented.
- **Manuel Serrano** gave a talk at the GG&JJ workshop in honour of Gérard Berry and Jean-Jacques Lévy, held in Gerardmer. He presented the *diffuse computing* at the Microsoft Software Summit in April. He presented the Hop programming language at Ircam in May. Manuel Serrano presented Hop at the meeting *Inria, Industrie* mobile initiative 2011.

9.2. Animation

- **Gérard Berry** is a member of the *Académie des sciences*, of the *Académie des technologies*, and the *Academia Europaea*. He is member of the steering committee of the *ANR*. He was the chair of the evaluation committee of University of Santander (12-16 Mars). He was the director of the scientific board of the computer science department of the *École Normale Supérieure* (May). He was a member of the scientific board of *IRCAM* (December). He was a member of *Franco-Taiwanais* award (Tapei 2011).

  Gérard Berry was the chair of the PhD jury of Stéphanie Delaune. We was a member of the HDR of Frédéric Boulanger, Supelec and Eric Madeleine, Nice. He was a member of the PhD jury of Marc Galceran, University of Barcelona.
- **Ilaria Castellani** was an examiner of the PhD thesis *Une étude dénotationnelle de la mobilité* by Joël-Alexis Bialkiewicz, UPMC, Paris (supervisor: Frédéric Peschanski).
- **Tamara Rezk** is a member of the programme committees of Bytecode, PLAS (co-chair with Sergio Maffeis), SAFA (co-chair). In December, she reviewed a grant proposal for the Estonian Science Foundation (ETF). She was a reviewer for TISSEC (journal) and MSCS (journal). In October, she was part of the jury in the thesis defense of Zhengqin Luo.
- **Manuel Serrano** is the coordinator of the ANR DEFIS project PWD. He served on the program committees of the following conferences:
  - SCHEME'11, 12th Workshop on Scheme and Functional languages.
  - DLS'11, 6th Dynamic Languages Symposium.
  - COORDINATION'11, 6th International Federated Conferences on Distributed Computing Techniques.

  Manuel Serrano was a member of the jury of the PhD Thesis of Benjamin Canou (Universities Paris 6 & 7) and Zhenqgin Luo (University of Nice). He was a member of the selection committee of PPS (Paris 7).

9.3. Teaching

- **Gérard Berry** gave the following conferences:
  - *Les inversions mentales de l'informatique*, Colloque Vivre Ensemble, organisé par le Conseil Economique, social et environnemental, 2 decembre 2011.
  - *Les inversions mentales de l'informatique*, University of Bordeaux, 24 novembre 2011.
  - *La révolution numérique dans les sciences*, Université populaire de la vallée de l'Eyrieux, 14 octobre 2011.
  - *Les inversions mentales de l'informatique*, club DANAE, session sur les systèmes d'informations des ressources humaines, 27 septembre 2011.
  - Table ronde *Préparés à Internet* à l'Ecole des Mines de Paris, 22 septembre 2011.
  - *Les bouleversements du numérique et les inversions mentales associées*, Ecole Centrale de Paris, 16 septembre 2011.
  - *Systèmes d'informations médicales*, CNAM, 23 juin 2011.
  - Conférence-débat *Attention, les digital natives arrivent*, Ecole de Paris du magagement, 7 juin 2011.
  - *Les inversions mentales de l'informatique, clefs de la révolution numérique* au Ministère de la recherche et de l'enseignement supérieur, 27 may 2011.
  - *Le monde devint numérique*, cycle de trois conférences organisées par Universciences à la Cité des sciences de La Villette, Paris :
    * *Les racines scientifiques du monde numérique*, 18 mai 2011,
    * *La révolution numérique dans les sciences*, 25 mai 2011,
    * *A la chasse aux bugs, la maladie du certain*, 8 juin.
  - *Révolution numérique et inversions mentales*, Séminaire d'Histoire et de Philosophie des Sciences, University of Montpellier, 26 april 2011.
  - *Les inversions mentales de l'informatique, racines de la révolution numérique*, Cycle de conférences *Sciences en question* du Collège de France à Tunis, organisé par l'Institut français de coopération, ambassade de France à Tunis, 11 april 2011. Live broadcast to the Universities of Sfax et Gabès and *palais des sciences* de Monastir. Recorded for a DVD.
  - *Informatique et société numérique* aux lycées français Gustave Flaubert de Tunis.
  - *Les inversions mentales du numérique*, Nancy-Université, 31 march 2011.
  - *Les inversions mentales du numérique*, les mercredis de l'Académie de Créteil, 9 mars 2011. Video record by the *rectorat de Créteil*.
  - Distinguished Lecture *The Mental Inversions that Root the Digital Revolution* University of Birmingham, 16 february 2011.
  - *Bouleversements numériques et inversions mentales* chez l'éditeur de logiciels financiers Murex, 1er february 2011.
  - *Les inversions mentales de l'informatique, racines de la révolution numérique*, La science à Sophia, Fondation Sophia-Antipolis, 1 february 2011.
  - *les aspects méconnus du numérique*, Assosciences Midi-Pyrénées, Toulouse, 12 january 2011.
  - *Pourquoi le monde et les sciences deviennent numériques* Jean Cocteau high school of Miramas, 4 january 2011.

- **Ilaria Castellani** taught 9h in the course "Programmation web diffuse" of the Paris 6 Master, UPMC, held together with Manuel Serrano and Tamara Rezk.

  From June to July 2011, she supervised the L3 internship "Prime information flows and compositionality of security properties or parallel programs" by Hadrien Batmalle.
- **Johan Grande** is giving first year algorithmics and programming (Python) tutorial/practical classes and supervising a third year programming project, both at Polytech'Nice (52 hours altogether).
- **Tamara Rezk** taught a course on provable cryptography in the master "Cryptographie et Securité" at University of Nice-Sophia Antipolis. She supevised two Phd students, Zhengqin Luo (who defended in october and stay as a postdoc at Inria) and José Santos. In February 2011, she taught a course on "Secure web programming", Paris 6 Master, UPMC.
- **Manuel Serrano** taught 9h in the course "Programmation web diffuse" of the Paris 6 Master, UPMC.

10. Bibliography

Major publications by the team in recent years

[1] G. BARTHE, T. REZK, A. RUSSO, A. SABELFELD. *Security of Multithreaded Programs by Compilation*, in "ESORICS", 2007, p. 2-18.

[2] G. BOUDOL, I. CASTELLANI. *Noninterference for Concurrent Programs and Thread Systems*, in "Theoretical Computer Science", 2002, vol. 281, n^{o} 1, p. 109-130.

[3] C. FOURNET, T. REZK. *Cryptographically sound implementations for typed information-flow security*, in "Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008", 2008, p. 323-335.

[4] M. SERRANO, E. GALLESIO, F. LOITSCH. *HOP, a language for programming the Web 2.0*, in "Proceedings of the First Dynamic Languages Symposium", Portland, Oregon, USA, October 2006.

[5] M. SERRANO. *Bee: an Integrated Development Environment for the Scheme Programming Language*, in "Journal of Functional Programming", May 2000, vol. 10, n^{o} 2, p. 1–43.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[6] Z. LUO. *Semantics and Security of Web Applications*, Université Nice Sophia Antipolis, Oct 2011.

Articles in Non Peer-Reviewed Journal

[7] G. BARTHE, P. D'ARGENIO, T. REZK. *Secure Information Flow by Self Composition*, in "Journal of Mathematical Structures in Computer Science", 2011, n^{o} 21, p. 1207-1252.

[8] G. BARTHE, D. PICHARDIE, T. REZK. *A Certified Lightweight Non-Interference Java Bytecode Verifier*, in "Journal of Mathematical Structures in Computer Science", 2012.

International Conferences with Proceedings

[9] G. BERRY, C. NICOLAS, M. SERRANO. *HipHop: A Synchronous Reactive Extension for Hop*, in "Proceedings of the PLASTIC'11 workshop", Portland, USA, Oct 2011.

[10] V. BONO, S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI. *A Reputation System for Multirole Sessions*, in "TGC", Lecture Notes in Computer Science, Springer, 2011.

[11] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI. *Information Flow Safety in Multiparty Sessions*, in "EXPRESS'11", EPTCS, 2011, vol. 64, p. 16–30, http://arxiv.org/abs/1108.4465v1.

[12] C. FOURNET, J. PLANUL, T. REZK. *Information-flow types for homomorphic encryptions*, in "ACM Conference on Computer and Communications Security", 2011.

[13] Z. LUO, T. REZK, M. SERRANO. *Automated Code Injection Prevention for Web Applications*, in "Proceedings of the first Conference on Theory of Security and Applications (TOSCA'11)", Saarbrücken, Germany, Lecture Notes on Computer Science, Apr 2011, vol. 6993, p. 186–204.

[14] B. SERPETTE, M. SERRANO. *An Interpreter for Server-Side Hop*, in "Proceedings of the DLS'11 symposium", Portland, USA, Oct 2011.

National Conferences with Proceeding

[15] P. ATTAR, F. BOUSSINOT. *Orchestration synchrone et au-delà*, in "Actes de la conférence Modélisation des Systèmes Réactifs (MSR'11)", Hermes, 2011, vol. 45, p. 77-92.

[16] E. CHAILLOUX, B. SERPETTE. *Séparation des couleurs dans un lambda-calcul bichrome*, in "Actes des journées JFLA", Carnac, France, 2012.

Conferences without Proceedings

[17] M. SERRANO. *HopTeX - Compiling HTML to LaTeX with CSS*, in "Online Proceedings of the Scheme'11 workshop", Portland, USA, Oct 2011.

Research Reports

[18] P. ATTAR, F. BOUSSINOT, L. MANDEL, J.-F. SUSINI. *Proposal for a Dynamic Script Language*, INRIA, May 2011, http://hal.inria.fr/hal-00590420.