Sécurité et Sûreté Informatique - Appel à projets 2007 (ANR-07-SESUR) **Fiche B : Description technique détaillée du projet**	
Acronyme du projet : SCALP	

Security of Cryptographic ALgorithms with Probabilities

Table des matières

Résumé du projet en français

Notre vie quotidienne dépend de façon croissante de la possibilité d'échanger des informations, et de le faire de façon sûre, protégée du risque de détournement par des personnes malintentionnées. Pour cela, il est nécessaire de mettre en œuvre des systèmes cryptographiques (protocoles et primitives).

Malheureusement, l'expérience montre qu'une mauvaise conception, ou une expression peu claire des propriétés et hypothèses de sécurité attendues conduisent à des attaques, et qu'il faut parfois des années avant que celles-ci soient découvertes et corrigées.

D'où l'adoption croissante de la *sécurité prouvable*, où on donne une définition rigoureuse des objectifs de sécurité et des démonstrations mathématiques que ceux-ci sont remplis. Par ailleurs, la complexité et la diversité des systèmes cryptographiques croît également. Il est donc largement admis qu'il n'est plus viable d'écrire ou vérifier manuellement des démonstrations cryptographiques (Bellare& Rogaway 2004, Shoup 2004, Halevi 2005) et qu'il faut développer des méthodes de vérification des systèmes cryptographiques *assistées par ordinateur*. Le but de ce projet est d'effectuer des progrès significatifs dans cette direction. Nous avons besoin à la fois de souplesse, en raison de la diversité des systèmes cryptographiques, et d'automatisation. Nous construirons donc nos outils autour de Coq.

Verrous scientifiques et technologiques

Notre projet devra s'intéresser aux points-clés suivants :

Langage probabiliste et sémantique pour les démonstrations cryptographiques Nous développerons un langage probabiliste permettant de décrire les systèmes cryptographiques à vérifier. *Il devra autoriser la description de jeux qui spécifient l'interaction entre des systèmes et leurs adversaires*, et sera donc plus complexe qu'un simple langage probabiliste. Ainsi, il doit permettre de spécifier un adversaire comme une boîte noire pouvant utiliser certains oracles. Il doit pouvoir modéliser des systèmes concurrents où l'adversaire peut affecter l'ordonnancement. Enfin, on devra pouvoir contrôler les ressources utilisées par un calcul.

Formalisation de générateurs aléatoires La correction des programmes aléatoires repose en pratique sur des propriétés des générateurs pseudo-aléatoires. Ces propriétés sont généralement présentées de façon abstraite et souvent admises implicitement. Nous définirons formellement les propriétés pertinentes pour les programmes probabilistes. Pour cela, nous nous intéresserons aux algorithmes de tatouage, pour lesquels il est bien connu que la mauvaise qualité du générateur ouvre la porte à certaines attaques.

Théorie de la démonstration Vérifier des systèmes cryptographiques requiert le développement d'un cadre théorique qui doit permettre :
- Un raisonnement de haut niveau sur les distributions définies par des programmes probabilistes.
- Des transformations de programmes préservant leur sémantique.
- Des transformations fondées sur des procédés cryptographiques.
- Du raisonnement asymptotique.

Résultats attendus

Au terme du projet, nous aurons un outil basé sur Coq pour démontrer la correction de systèmes cryptographiques. Nous vérifierons son utilité en l'appliquant dans trois domaines importants : les protocoles d'échange de clés, les systèmes de type pour le contrôle computationnel des flux d'information et le contrôle d'intégrité, et le tatouage. Contrairement aux outils fondés sur le modèle de Dolev-Yao, nos démonstrations seront fondées sur le modèle de la théorie de la complexité et nous pourrons nous intéresser aux protocoles de groupes. En outre, notre outil sera ouvert, au sens où il sera raisonnablement facile de s'intéresser à de nouvelles primitives, problème qui semble difficile pour les outils symboliques.

Summary in English

Our day-to-day lives increasingly depend upon information and our ability to manipulate it securely. That is, in a way that prevents malicious elements to subvert the available information for their own benefits. This requires solutions based on cryptographic systems (primitives and protocols).

However, no matter how carefully crafted cryptographic systems are, experience has shown that effective attacks can remain hidden for years. This may be caused by poor design or often unclear and poorly defined security properties and assumptions.

Therefore, provable security, where new systems are published with a rigorous definition of their security goals and a mathematical proof that they meet their goals, is being increasingly advocated. While the adoption of provable security significantly increases, the complexity and diversity of designed systems tend to increase too. Hence, it is largely agreed on that the point has been reached where it is no longer viable to construct or verify cryptographic proofs by hand (Bellare& Rogaway 2004, Shoup 2004, Halevi 2005) and that there is a need for *computer-aided* verification methods for cryptographic systems. The goal of this project is to achieve a major step towards building automated tools for the verification of cryptographic systems. In order, to reconcile generality, imposed by the high diversity of cryptographic systems, and automation, we shall build our tools upon Coq.

Challenges to be addressed

The success of our project will depend on mastering the following key challenges.

Probabilistic language and semantics for cryptographic proofs We need to develop a probabilistic programming language that allows to describe the cryptographic systems to be verified. It has to be more complex than a simple probabilistic language since *it must allow the description of games that specify the interaction between adversaries and the systems.* Therefore, it should allows us to specify an adversary as a black-box that has access to some oracles ; it needs to express concurrent systems where the adversary may affect scheduling ; and it should be possible to characterize resource-bounded computations.

Formalization of random generators Most properties of random programs rely on good properties of pseudo-random generators. These properties are generally presented in a very abstract way and are generally implicitly admitted in proofs of probabilistic algorithms. We shall define formally the relevant properties for a probabilistic program. We will focus on watermarking algorithms for studying this area.

Proof theory Verification of cryptosystems needs the development of a specific verification theory that has to include the following proof activities :
– High-level reasoning about distributions defined by probabilistic programs.
– Semantic preserving program transformations.
– Cryptography-based program transformations.
– Asymptotic reasoning.

Expected results

By the end of the project we expect to have a Coq-based tool for proving correctness of cryptosystems. We will demonstrate its usefulness by considering three major areas : key-exchange protocols, computationally sound type systems for non-interference and data integrity, and watermarking. In contrast to existing Dolev-Yao based verification tools, we will be able to treat group protocols and have complexity-theoretic proofs. More importantly, the tool will be open in the sense that it should be reasonably easy to consider new primitives. This does not seem easily achievable for symbolic tools.

Introduction

Our day-to-day lives increasingly depend upon information and our ability to manipulate it securely. That is, in a way that prevents malicious elements to subvert the available information for their own benefits. This requires solutions based on cryptographic systems (primitives and protocols). Shannon's early work in the 1940's already showed that perfect (unconditional) security is difficult to achieve in practice. Three decades later (1976), Diffie and Hellman invented public-key cryptography, coined the notion of one-way functions and discussed the relationship between cryptography and complexity theory. Shortly after (Rabin 1979), the first cryptosystem with a reductionist security proof appeared. The next breakthrough towards formal proofs of security is the adoption (notably by Goldwasser, Micali, Goldreich and Rivest) of computational theory for the purpose of rigorously defining the security of cryptographic schemes. In this framework, a system is *provably secure* if there is a poly-time reduction proof from a hard problem to an attack against the security of the system. A user of this framework must specify how an arbitrary, probabilistic, poly-time adversary attacker can interact with legitimate users and must specify what the attacker should achieve in order to break the cryptosystem. The provable security framework has been later refined into *the exact (also called concrete) security framework* where better estimates of the computational complexity of attacks is achieved. Beyond the verification of cryptographic systems and constructions, research in the area of provable and exact security investigates the relationship between different security notions and security models.

Provable cryptography has become a very active field of research well represented in top cryptographic conferences, and cryptographic schemes, including standards, are increasingly supported by correctness proofs. Yet, there is a problem with cryptographic proofs, as expressed by Shai Helevi in [44] : *Do we have a problem with cryptographic proofs ? Yes, we do. The problem is that as a community, we generate more proofs than we carefully verify (and as a consequence some of our published proofs are incorrect).* Alexander Dent [39] has even stronger words : *... cryptographers are only just beginning to develop the mathematical rigor that they need in order to be able to produce algorithms and protocols in which one can have true confidence.* J. Stern et al. write in [81] : *...the use of provable security is more subtle than it appears, and flaws in security proofs themselves might have devastating effect on the trustworthiness of cryptography.* Unfortunately, flaws in security proofs are not that seldom (cf.[81, 28]).

Our aim is to develop general computer-aided tools for verifying cryptographic systems with strong correctness guarantees. Our tools should have the following characteristics :

1. Support the provable and exact security frameworks.
2. Be reasonably automated.
3. Be applicable to realistic systems. We target three application areas : password-based group key exchange protocols, computationally sound type systems for information flow and watermarking algorithms.
4. Put emphasis on proof checkability and certification. This is in order to increase the trustworthiness in developed proofs.

We shall build our tools on top of the Coq proof assistant, which offers the necessary mathematical expressivity to reason about probabilities, programming language semantics, polynomials, etc, and which depends on a small and well-understood logical core for having strong correctness guarantees.

Challenges to be addressed

The success of our project will depend on mastering the following key challenges.

Probabilistic language and semantics for cryptographic proofs We need to develop a programming language that allows to describe the cryptographic systems to be verified. As cryptographic systems usually involve randomness and probabilistic behavior the language has to be probabilistic. But the language has to be more complex than a simple probabilistic language as *it must allow the description of games that specify the interaction between adversaries and the systems.* Therefore we shall address the following issues :

1. An adversary can be seen as a black-box that has access to some oracles. The latter may depend on secret keys. Hence, the language has to include variables that refer to unspecified code that

has access to oracles. The language has also to include means for specifying access policies to the system variables.

2. Cryptosystems may involve concurrency as many agents and even many instances of the same agent may be running the system. Clearly, interleaving-based semantics is not appropriate for security proofs. There are essentially two possible views : the system is a set of communicating processes and the adversary controls the scheduling of both processes and messages. Or, the system consists of one sequential process, namely the adversary, who has access to some state-full oracles. Obviously, the choice will largely affect the shape of the proofs we will get as well as the degree of automation we can reach.
3. Since we need to support reductionist proofs that involve the construction of (possibly) poly-time probabilistic programs, the question of how such resource bounded computations can be characterized affects the reachable degree of proof automation. Indeed, reductionist proofs involve the proof obligation that consists in showing that the reduction is bounded under the assumption that adversary is bounded.

Formalization of security properties A key issue toward building and popularizing automated tools for the verification of cryptosystems is the development of canonical definitions of widely used security properties. In particular, we should draw a clear picture of the landscape of existing computational non-interference definitions. The same holds for properties we expect a group key exchange protocol or the properties a watermarking system should satisfy. More generally, we need to define and study observational equivalences of probabilistic systems that include adversaries.

Proof theory Verification of cryptosystems needs the development of a specific verification theory that has to include the following proof activities :

- **Invariants.** We need to develop abstract high-level reasoning about distributions defined by probabilistic programs. Typically, we need to prove invariants that state the independence of some random variables. For instance, we want to be able to compositionally show that a given probabilistic program preserves indistinguishability of distributions.
- **Semantics preserving program transformations.** As stated by Shai Halevi [44] many steps in a cryptographic proof boil down to "code motion" or more generally to program optimization. We should, however, not forget how many heads has the beast we are dealing with : probabilistic behavior, unspecified code, oracles, concurrency etc... Thus, we need to specify and prove correctness of code optimization for cryptoprograms.
- **Cryptography-based program transformations.** A different kind of step in a cryptographic proof are reduction proofs. In such steps, we want to establish a property by reducing from a cryptographic or number theoretic assumption such as prime number factoring, RSA, CDH or DDH. At the heart of such proof step, we have the construction of a probabilistic program that uses a set of programs as black-boxes. One of the proof obligations that need to be discharged is that the constructed program runs in poly-time. Consequently, complexity definitions and reasoning need to be captured by our tools and formalizations.
- **Asymptotic reasoning.** An other feature that distinguishes cryptoproofs is that the reasoning about probabilities may be asymptotic. A typical security property states that for any adversary the probability of an attack is ultimately negligible. This is an asymptotic notion. An alternative to asymptotic is to work within the concrete (also called exact) security framework. The aim is here to give more precise bounds on the probability of attacks. This is indeed preferable. But asymptotic reasoning might be useful first to get a simple proof of correctness that can be refined into a concrete proof. We need to formalize both types of reasoning and we need to develop proof strategies that allow to refine an asymptotic proof into an exact one without repeating all proof steps.

Expected results

By the end of the project we expect to have a prototype of a Coq-based tool for proving correctness of cryptosystems. We will demonstrate its usefulness by considering three major areas : key-exchange protocols, computationally sound type systems for non-interference and data integrity and watermarking.

In contrast to existing Dolev-Yao based verification tools, we will be able to treat group protocols and have complexity-theoretic proofs. More importantly, the tool will be open in the sense that it should be reasonably easy to consider new primitives. This does not seem easily achievable for symbolic tools. The following results, mandatory for the development of our targeted tool, are interesting and important on their own :

- A probabilistic language for cryptosystems and cryptoproofs with formalized semantics and reasoning.
- A library of proved program transformations based on code optimization techniques.
- Rigorous definitions and reasoning of observational equivalence relations for probabilistic programs.
- A library of proofs of well-known properties about measures of classical distributions (Binomial,. . .).
- A framework for proving probabilistic watermarking algorithms, based on libraries above.
- A library of lemmata and tactics that support reductionist proofs.
- A Coq-proved type system for non-interference and data integrity for a programming language that includes random number generation, block ciphers and password-based encryption. The correctness we seek is complexity-theoretic.
- A library of results about random generators.

Contexte et état de l'art / Context and state-of-the art

Certification of cryptographic systems

There are two competing, and in fact complementary, approaches to the verification of cryptographic systems and protocols. In the so-called formal, symbolic or Dolev-Yao model [40], data is specified using an abstract data type (algebraic specification) and are manipulated by systems and *adversaries* according to this abstract data type. In other words, the abstract data type gives an abstract specification of the cryptographic primitives and of the computational power of the adversaries that try to break the security properties. In this model, security properties are expressed as safety properties or using observational equivalences.

In the complexity-theoretic model (e.g. [42, 20]), also called the computational model, the adversary can be any poly-time probabilistic algorithm. That is, data manipulation is not restricted to programs that are sequences of operations taken from a fixed finite set of operations but can be performed using any poly-time probabilistic algorithm. Moreover, in this model security properties are expressed in terms of the probability of success of any poly-time attack. Attacks are usually defined in terms of probabilistic games, where the adversary has access to some oracles and wins the game, if he/she correctly answers a question. A typical question is to distinguish between a data computed by the cryptographic system and a randomly chosen value.

While the complexity-theoretic framework is more realistic and gives stronger security guarantees, the symbolic framework allows for a higher level of automation. Because of this, effort has been spent during the last years in relating both frameworks with the goal of getting the best of two worlds [2, 1, 66, 65, 12, 57, 47, 27, 53, 34, 52, 62]

It is fair to say that the symbolic approach has reached some limits. Indeed, many cryptographic primitives, such as commutative encryption, blind signature and the xor operation, require symbolic modeling that relies on involved equational theories or combinations thereof for which either we don't have verification algorithms at all or algorithms that cannot be easily implemented [56, 38, 19]. And in case, such algorithms exist or would be implemented, we would need to prove their soundness with respect to the complexity-theoretic model. Some negative results [43] may indicate that this is impossible or at least difficult for some cryptographic primitives. An other limitation of the symbolic methods is that they do not support the *concrete security framework* which consists in analyzing the computational complexity of attacks providing more precise estimates than polynomial equivalence.

A formal security model consists of definitions : it must specify how an arbitrary poly-time probabilistic adversary interacts with the honest agents (the cryptosystem), and it must state what the attacker should achieve to break the system. There are two general security model styles. In the *game-based* approach [42], a model consists of an attacker and a challenger. The attacker is interacting with the challenger and wins the game when he outputs a value that satisfies a defined winning condition. In this approach, security properties are defined in terms of the probability to win the game. Widely accepted games are Indistinguishability games. The other style of security models is *simulation-based* (see [71, 27]). In this approach, there are two worlds. In the first world, the adversary interacts with an environment (the users of the cryptosystem) and the cryptosystem. While in the second world the cryptosystem is replaced by an idealized system that (miraculously) implements an idealized version of the sought functionality. The cryptosystem is then called secure, if no one having access to the adversary's behavior can tell in which world it takes place. While simulation-based security proofs are stronger, there are security functionalities that cannot be proven in this model. Until recently, both approaches suffered from the lack of tools that support the development of proofs. Bruno Blanchet developed a dedicated tool that supports concrete security proofs within the game-based approach [25, 26]. Also recently, Peeter Laud developed a protocol analyzer [57] that works with the Backes-Pfitzmann-Waidner cryptographic library [71, 13]. The tool does not directly give a complexity-theoretic proof but rather a symbolic one. However, combined with the complexity-theoretic soundness of the library [13, 12, 14] this gives a complexity-theoretic guarantee.

Maybe the closest to our project is the work carried by Gilles Barthe and Sabrina Tarento who were among the first to provide machine-checked proofs of cryptographic schemes without relying on the perfect cryptography hypothesis. More concretely, the team has provided formal models of the Generic Model, or GM for short, and the Random Oracle Model, or ROM for short, in the Coq proof assistant, and used this formalization to prove hardness of the discrete logarithm [16], security of signed ElGamal encryption against interactive attacks [18], and of Schnorr signatures against forgery attacks [83].

Another interesting work to mention is the Hoare-style proof system proposed by R. Corin and J. Den Hartog for game-based cryptographic proofs [33]. Yet, there is no computer-assistance for the developed logic. Moreover, it is based on “pure” Indistinguishability not taking into account computation branches where games are distinguishable, i.e., bounding the probability of this to happen. In [37], Datta et al. present a computationally sound compositional logic for key exchange protocols. There is, however, no proof assistance provided for this logic.

Certification of probabilistic programs

There are relatively few tools for formally reasoning on probabilistic programs. A first category of tools is model-checkers analyzing probabilistic transition systems. They are used for the verification of distributed algorithms.

The group of M. Kwiatkowska in Birmingham has designed a probabilistic model-checker PRISM [55], which uses Markov chains as underlying model and a probabilistic temporal logic for queries. Reasoning in this framework is automatic but requires complex computations. It has been used for analyzing a large number of distributed systems.

Another approach for model-checking probabilistic systems is based on statistical analysis : it provides an approximated estimation of the probability for a property to hold on the system, this is the case for the model-checkers VeStA [77] or APMC [46].

Probabilistic rewriting theories are also a popular framework for modeling and analyzing randomized systems. Probabilistic extensions of rewriting logics are used for quantitative analysis of properties. PMaude is an extension of the rewriting system Maude with probabilistic rules [4].

PRISM can also be used for the verification of programs expressed in other formalisms such as probabilistic rewriting systems [45].

A second category of tools is based on a mathematical reasoning on a model of probabilistic programs in a proof assistant.

The language of probabilistic guarded commands proposed by A. McIver and C. Morgan [67] has been developed in the proof assistant HOL [51]. They formalized the syntax and the semantics of the language as well as a generator of so-called *pre-expectations* which evaluates the expectation of a function with respect to the measure associated to the program. HOL tactics provide partial automation of proofs. This framework handles non-deterministic as well as probabilistic choice but very few examples have been developed in that system : the paper only presents Rabin’s mutual exclusion algorithm.

J. Hurd [49, 48, 50] shows how to model and prove properties of randomized (but deterministic) programs in the HOL proof assistant using a monadic transformation of programs, where the author assumes access to an infinite sequence of independent coin flips. This modeling of programs is appropriate for simulation but reasoning on the quantitative properties requires to model the probability space of infinite sequences of random bits. A typical example developed using this approach is the Miller-Rabin Probabilistic Primality Test.

In [10], we proposed a method for representing randomized functional programs in the Coq proof assistant using an interpretation of programs as measures. This representation is suitable for reasoning on quantitative properties of randomized programs, even using general recursion. Different primitives for random constructions can be chosen, discrete or continuous. Reasoning involves algebraic manipulations that can be done interactively in the Coq system.

Certification of watermarking algorithms

Watermarking [36] consists in the robust and non-obtrusive insertion of a *mark* in a digital media (*data*). The mark may be for example a copyright information [79]. A watermarking system is a pair of (generally) probabilistic programs M and D. M inserts the mark by modifying bits. The modified bits are selected and altered by a pseudo-random generator with a specific secret seed : the *key.* D performs, given the key, the detection of the mark by verifying the chosen bits. Watermarking can be seen as a particular kind of symmetric cryptographic process : the mark is encrypted with the help of the data and the secret key, which D must know to detect the mark.

A watermarking system must at least be *robust* : without knowing the key, it is hard to alter the data in such a way that 1) D is unable to detect the mark, and 2) the data stays useable, i.e. the "distance" between the genuine data and the altered data remains within an acceptable range (a parameter of M). A watermarking system is *secure* if it is difficult to retrieve information about a key k from one (or several) k-marked data.

Significant work has been made on formalizing and proving formally watermarking properties. Among many others Adelsbach et al. use two party games to model the information leaked through the marked data [7], Cousot and Cousot [35] make use of abstract interpretation to formalize *software watermarking* and Barni et al. formalize watermarking in an operational way [63]. However to our knowledge there has not been any attempt to mechanize such proofs. Our formalization of watermarking will build upon Barni's work and use the framework of WP1. We may also use game based proofs as Adelsbach et al. and as explained in WP3 to formalize the *security properties* of watermarking algorithm.

Project positioning w.r.t. other projects funded by ANR

To the best of our knowledge, there is no project (nationally or internationally) aiming at providing general computer-aided tools for the verification of cryptographic systems. There are, however, a number of related projects with potential synergy.

FormaCrypt [1] is the project that comes closest to ours. While much of the effort in FormaCrypt focusses on the computational soundness of symbolic methods, Bruno Blanchet developed a tool, CryptoVerif, for game-based computational proofs. The main differences in the focus of our project and FormaCrypt is that we put emphasis on certification of proofs (i.e. we intend to produce independently verifiable proofs of security), and that we aim for case studies that are far beyond the current capabilities of CryptoVerif, and necessitate falling back on general purpose theorem provers. In the longer term, it seems natural to integrate CryptoVerif or a similar tool as a procedure for discovering sequences of games whose correctness can be established using our tools. WP3 takes preliminary steps in this direction.

PARSEC [2], in which EVEREST-INRIA is involved, is a project that focuses on language based security, whose goal is to counter application-level attacks through appropriate language-base mechanisms, in particular type systems. PARSEC has a strong focus on information flow [73], and on declassification [74]. One proposed application of the SCALP project is also to develop type systems for information flow, in the context of programming languages with cryptographic primitives—and where declassification occurs through the release of encrypted data. Thus, there are potential synergies with the ANR project PARSEC, in which EVEREST-INRIA is involved, and which has a strong focus on information flow ; nevertheless, PARSEC only focuses on possibilistic (non-computational) settings ; in particular, the programming languages considered in PARSEC do not include cryptographic primitives.

Other projects with potential synergies include the ARA project CompCert on certified compilation.

Project positioning with respect to the call "Sécurité et Sûreté Informatique"

The call identifies four axes :

1. Information systems security (Sécurité des systèmes d'information)
2. Information systems safety (Sûreté des systèmes informatisés)
3. Trust justification (Justification de la confiance)
4. Societal aspects of information security (Aspects sociétaux de l'informatique sécuritaire)

The SCALP project falls in the two first items and is directly related to the third.

[1] `http://www.di.ens.fr/~blanchet/formacrypt/`
[2] `/http://moscova.inria.fr/~zappa/projects/parsec/parsec.html`

Partenaires / Partnership

The consortium is composed of five teams with an outstanding experience in formal proofs and their applications to mathematics, programming languages and security. The teams have successfully collaborated in the past on a number of topics, many of which are directly relevant to the project. In particular, ProVal-LRI, EVEREST-INRIA, and CPR-Cédric contribute to the development of the Coq proof assistant, and have previously collaborated in the formalization of programming languages, and of tactics ; they share a strong expertise in the design of tactics, and especially reflective tactics. Besides this, ProVal-LRI and Plume-LIP have collaborated on the study of probabilistic programs and have developed a formalized library of probabilities that will serve as a starting point for the work in this project. Finally, EVEREST-INRIA and DCS-VERIMAG have initiated an informal collaboration on the topic of formalized game-based proofs, building upon previous work on the formalization of ideals cryptographic model at EVEREST-INRIA and on the soundness of the formal model vs. the computational model at DCS-VERIMAG.

The unique blend of competences of the teams, together with their continued history of successful collaborations, put the consortium in the ideal position to complete the proposed research and have a strong impact in the use of proof assistants in cryptography.

DCS-VERIMAG

The research will take place in the DCS (Distributed and Complex Systems) team of Verimag. The DCS team (`http://www-verimag.imag.fr/~async`) is composed of 11 permanent researchers, 10 PhD students, 4 post-docs and 1 engineer.

The expertise of the team is in the area of semantics and systems verification. The team research is organized according to three axes : 1.) Methods and tools, including model-checking and static analysis, for real-time embedded systems, 2.) Automated verification of pointer programs and 3.) Formal security including verification of cryptographic systems, testing of security policies and Smartcard applications certification. The background of the team is program and reactive system semantics, abstract interpretation and model-checking, formal languages and automate theory. More recently, the team competences have been enriched with type theory and theorem proving. The team has long experience in tool development and software dissemination (8 tools are distributed [3]).

R. Janvier, who is currently a postdoc in the Everest project, and L. Mazaré have defended their thesis in the area of the verification of cryptographic protocols and more specifically the link between symbolic proofs and proofs in the complexity-theoretic approach.

The main researchers involved in the current project will be Cristian Ene, Jean-François Monin, Judicaël Courant and Yassine Lakhnech. These are members of the security group of the DCS team.

The DCS team is involved in the following projects related to security :

- ACI POTESTAT (2004-2007) : Formal modelling of security policies and test case generation.
- RNTL EDEN2 (2006-2009) : Smartcard certification.
- RNRT Politess (2006-2009) : Formal modelling of security policies and test case generation.

and coordinates the Network of Excellence Artist on Embedded Systems (2002-2008).

EVEREST-INRIA

The research will take place in the EVEREST team at INRIA Sophia-Antipolis. The team is composed of 3 permanent researchers, 6 Ph.D. students, 2 post-docs and 1 engineer.

The EVEREST team shall be in charge of WP3, and shall also contribute to WP1-WP2-WP4. The main contributors to this project shall be Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin, and a Ph.D. student recruited to work on certified cryptographic systems and information flow analyzes.

The project concentrates on increasing reliability and security of mobile and embedded software. This is achieved by developing and applying formal methods and language-based techniques, covering both platform and application level. The project's privileged application domain ranges from trusted personal devices, such as mobile phones and smart cards, to ubiquitous computing.

[3] `http://www-verimag.imag.fr/index.php?page=tools&lang=en`

The project focuses on the following research themes : program verification ; language-based security (including information flow) and Proof Carrying Code ; machine checked programming language semantics ; machine-checked cryptography ; and foundations of proof assistants.

Since 2003, the team is involved in the machine-checked formalization of idealized models, and has produced some of the earliest machine checked proofs of correctness of cryptographic systems that do not make the perfect cryptography assumption. The research was mostly carried in the context of the Ph.D. work [82] of Sabrina Tarento. Since 2006, the team has begun investigating code-based cryptographic proofs, which is at the heart of WP3.

In addition, the team has been active in the area of language-based security, and in particular in the area of information flow type systems [72], that is directly relevant to WP4, and has a strong background in program equivalence and compiler certification, which is directly relevant to WP2.

The EVEREST project is involved in the following projects :

– FET Integrated Project Mobius (Mobility, Ubiquity and Security), 2005-2009 ;
– ANR Sécurité PARSEC (Parallélisme et Sécurité), 2006-2010.

Gilles Barthe, Benjamin Grégoire and Santiago Zanella are also involved in the INRIA-Microsoft Research Joint Center.

Plume-LIP

This research will take place in the Plume team of the *Laboratoire d'Informatique du Parallélisme*, at the École Normale Supérieure de Lyon.

The Plume team (`http://www.ens-lyon.fr/LIP/PLUME/`) is composed of 5 permanent members and 3 PhD students.

The expertise of the team include the semantics of programming languages and systems in their many aspects, and formal proofs with concerns in formal methods and verification techniques.

The main permanent researcher involved in the current project will be Philippe Audebaud. He is already collaborating with Christine Paulin-Mohring on the design of a probabilistic functional language, and the development of tools for proving properties in this area.

The Plume team is involved in

– MoDyFiable : (2006-2008) Mobilité Dynamique Fiable.
– EmSoC-Recherche : project from cluster "Informatique, Signal, Logiciel Embarqué" (ISLE), a Rhône -Alpes french region based project.

ProVal-LRI

The research will take place in the ProVal project, a joint project between Laboratoire de recherche en Informatique (UMR 8623 CNRS, Université Paris-Sud) and INRIA Futurs.

The ProVal project (`http://proval.lri.fr`) is composed of 7 permanent researchers, 7 PhD students, 3 post-docs and 1 engineer.

The expertise of the team is in the area of proofs of programs. It consider the deductive verification of functional as well as imperative programs. Proofs are done either interactively using the Coq proof assistant or automatically using dedicated tools for solving proof obligations generated by program verification. The team as a strong experience in the development of tools for verification : it participates to the Coq (`http://coq.inria.fr`) development team ; it develops the CiMe [31] toolbox for analyzing rewrite systems. More recently it designed the Why tool (`http://why.lri.fr`) for analyzing imperative programs with dedicated front-ends (Caduceus and Krakatoa) for dealing with C and Java programs.

The main permanent researcher involved in the current project will be Christine Paulin-Mohring. She already developed a large Coq library for modeling randomized programs [69] which is available as a Coq contribution. On that subjected she collaborated closely with Ph. Audebaud in the Lyon site.

The ProVal project is involved in national projects :

– A3PAT Assister automatiquement les assistants de preuve avec des traces (ANR Programme Blanc 12/2005-12/2008)
– CAT C Analysis Toolbox (ANR RNTL 01/2006-12/2008)
– CERPAN Certification de Programmes numériques (ANR Programme Blanc, 12/2005-12/2008)
– Plate-forme de Confiance (FCE 01/2007-06/2009, competitive cluster System@tic)

ProVal also participates to an European coordination action TYPES on Type Theory.

C. Paulin-Mohring is mainly involved in the PFC project.

CPR-Cédric

The research will take place in team CPR of the *Centre d'Études et de recherches en Informatique du CNAM* (CEDRIC). The CPR team (`http://cedric.cnam.fr`) is composed of 10 permanent members, 2 PhD students and 3 post-doc students. The expertise of the team includes formal methods in relation with the development of certified programs. It gathers COQ experts whose skills include modelling, proof tactics, proof terms, etc. The team develops in particular C*i*ME (joint work with ProVal team), which produces certificates of complex properties (termination...) for Coq (A3PAT project (`http://www.ensiie.fr/~urbain/a3pat`). The team is also involved in the development of Coq.

The main permanent researchers involved in the present project will be : Pierre Courtieu and Xavier Urbain. They are already working on the proof of a small watermarking system using the library for modeling randomized algorithm developed by C. Paulin-Mohring (BQR project).

P. Courtieu works in relation with the Coq proof assistant, in particular by defining proof tactics aimed at certifying JavaCard platforms [17] in the (EU-IST) VERIFICARD project. More generally he works on reasoning over functional specifications. He is also working in the A3PAT project.

X. Urbain is co-author of the second version of C*i*ME [31, 32] and Krakatoa [61] (certification of JML annotated Java programs) in the (EU-IST) VERIFICARD project. He is the coordinator of the (ANR) A3PAT project.

Courtieu is involved in the following projects :
- A3PAT
- CompCert
- SSURF

Urbain is involved in the following projects :
- A3PAT

Courtieu and Urbain work closely with David Gross-Amblard (University of Dijon, France) on the problematic of certification of probabilistic watermarking algorithms. Gross-Amblard works on probabilistic algorithms, and in particular on watermarking constrained data (ACI Sécurité TADORNE). Certifying this kind of algorithms is one of the ultimate goal of the case studies of the present project.

Organisation et management du projet / Project organization and management

The governance of project is structured as follows :

- Project management : Yassine Lakhnech
- Scientific Coordination : the scientific board consists of the project manager, team coordinators and WP-leaders.
- Responsible for dissemination : Judicaël Courant

Each team has a coordinator :

- DCS-VERIMAG : Yassine Lakhnech
- EVEREST-INRIA : Gilles Barthe
- Plume-LIP : Philippe Audebaud
- ProVal-LRI : Christine Paulin-Mohring
- CPR-Cédric : Pierre Courtieu.

Each workpackage has a leader :

- WP1 : Philippe Audebaud
- WP2 : Christine Paulin-Mohring
- WP3 : Gilles Barthe
- WP4 : Judicaël Courant
- WP5 : Pierre Courtieu.

WP-leaders are responsible for coordinating the corresponding workpackage, for planning WP-meetings, for monitoring the progress and interaction within the workpackage and finally, for meeting the deliverables and deadlines.

Kick-off meeting. The SCALP project will start by a kick-off meeting with all participants. Each WP leader will present a detailed program for the work-package he is responsible of. It is the project manager's responsibility to organize the kick-off meeting.

Plenary meetings and WP meetings There will be two plenary meetings per year ; one of which is a review meeting. It is the project manager's responsibility to organize the plenary meetings.

Review meeting. The scientific board prepares the annual meeting with all participants. The aim of these meetings is to assess the progress achieved by each WP and the project globally and to decide about necessary adaptations of the research agenda in order to meet the goals of the project.

Communication. The dissemination coordinator has the responsibility for creating a web site of the SCALP project. It will be used both for the communication within the project and for the dissemination of results. All internal information (activity reports, minutes of meeting, etc.) will be accessible for all participants. Moreover all publications will be available. We also plan to present our results in well recognized international conferences. The latter is the responsibility of all project members.

Project duration The project is planned over 48 months. This duration is necessary given that : the project includes ambitious theoretic work, languages and tools development and also non-trivial case studies. On the other hand, we plan to have after 24 months a prototype verification tool. We also plan to have by that time an evaluation of the tool on substantial case studies. Clearly this checkpoint at 24 months excludes should allow us to adjust our research agenda, if necessary.

Structure du projet – Description des sous-projets / Structure – Work-packages

We aim at providing general tools and methods for the logical analysis of properties of cryptographic systems involving random computations, which we shall extensively test on three non-trivial case studies. The infrastructure will be based on the Coq proof assistant for several reasons :

- The Coq theory based on higher-order logic is an adequate framework for modeling abstract notions, semantics of programming languages and advanced mathematical results in analysis.
- The Coq proof architecture offer strong guarantees on the correctness of proofs and allows to combine interactive with automatic proofs using tactics.
- The participants have a common expertise using this system.

In order to facilitate case studies and later ease the adoption of our tools by scientists with a lesser expertise in Coq, e.g. cryptographers, we shall develop dedicated tools customized to our application domains, while allowing the possibility to fall back on the generality of Coq for specific tasks not covered by our tools.

Therefore, we aim at designing and implementing a suitable front-end language (WP1), developing a dedicated environment for cryptographic proofs (WP2) and testing this language and this environment (WP3, WP4, WP5). Although the latter workpackages depend on the former, the expected workflow is certainly not linear ; instead the workflow should be iterative as we expect workpackages 3, 4 and 5 to foster WP1 and WP2.

WP1 Designing a probabilistic language

Leader : Plume-LIP *Participants : All for the definition of the characteristics of the language, Lyon and Orsay for its implementation*

This workpackage aims at defining both a language and its semantics for representing random computations that will be used for modeling randomized algorithms.

In order to provide a sufficient coverage of the formalisms used to model randomized algorithms, we intend to support various input languages, or successive versions of them, including at least a functional language that is convenient for reasoning, and an imperative language that is most commonly used in practice, e.g. in game-based proofs. To this end, we shall design an intermediate, or pivot language ; in this process, the eventual choice will be chiefly influenced by the applications studied in the other packages. In particular, we retain the following issues as essential :

- We shall decide how to represent “open programs”, that is programs involving oracles or adversaries and how these oracles or adversaries access to the program variables.
- We shall provide association lists or arrays as these are ubiquitous in cryptographic algorithms.
- We shall decide which amount of imperative features and distributed computations are necessary.
- We shall consider the possibility to characterize polynomial computations for programs written in this language.

The design could be influenced by the following considerations :

Imperative features The Why toolkit [41] is based on a monadic transformation on a simple, effective imperative language, which captures (imperative) effects over functional calls. Although this is done automatically from analyzing the source code, our current context could require a more explicit handling of such imperative variables, yet following the Why approach.

Concurrency Besides the operational prospect which would influence the language design, we are concerned with the fact concurrency involves non determinism of a quite different flavor as randomness does. In this aspect, adapting or generalizing McIver and Morgan approach (see e.g. [64]) could show beneficial in the process. Another approach would be to consider that the adversary controls scheduling, either by considering the scheduler as an unknown part of the program, or by considering the honest participants as an API in which the adversary can make arbitrary calls.

Dealing with distributions in the syntax When random primitives are added to a language, it is still possible to transform any program into some purely functional program. This is done as for imperative features with the help of the monadic approach. However, the transformation applies to the whole term, even to a pure deterministic one. Enhancing the syntax to allow for measures as first class citizen is feasible, along the work done by Davies and Pfenning [70]. The probabilistic functional language λ_O [68] might suggest elements of design in this direction. The $\mathbb{R}$ml language we proposed in [11] can be adapted as well.

Simulation concerns We shall also consider the possibility to simulate the behavior of our programs in addition to formal verification. If a functional design were to be retained, this point would advocate choosing λ_O over $\mathbb{R}$ml, as the latter was developed with proof generation in mind ; nevertheless, our technique can be brought to the former at no further expense, depending on expressiveness purposes.

A specification language will be designed which should be general enough to allow simple specification of security properties but restricted enough in order to facilitate automation.

We shall define different semantics for our language : An operational semantics will be suitable for simulation while a semantics in terms of random variables will be suitable for quantitative analysis or specification of equivalence between programs.

These semantics will be formally defined in Coq and will be used as a basis for implementation of tools for reasoning on probabilistic algorithms and cryptographic systems.

Milestones

This workpackage is based on previous work by the partners : a functional randomized language developed in Orsay and Lyon, a simple imperative language for non-interference analysis at Sophia-Antipolis. In this project, we shall propose a uniform framework with a core functional language for semantics analysis and specific input languages well-adapted for cryptographic analysis. These languages will be designed in the project iteratively by developing new features and confronting them to the case studies.

– D1.1 : T0 + 6 - formal definition of the simple core langages (programs and specification)
– D1.2 : T0 + 12 - first prototype implementation of the simple core language and its semantics
– D1.3 : T0 + 24 - evaluation of the prototype on case studies
– D1.4 : T0 + 36 - final core language and its formalization

WP2 Proof environment for security programs

Leader : ProVal-LRI *Participants : all*

This workpackage aims at developing a proof environment for security programs. This involves the development of dedicated proof tactics as well as specific libraries providing generic results for particular methods involved in the verification of cryptographic systems.

Certifying probabilistic programs in a security setting involves proofs of invariants over distribution of the program states, proving transformation in order to reduce games to known difficult problems and automation of these transformations.

Invariants

Invariance properties play a central role in Computer Science ; in our context, they are required by game-base proofs to justify program transformations to be applied, and to apply stability criteria, and to prove asymptotic properties of these programs.

We shall develop a formalized theory of invariants along two major prospects. Correctness of program transformations and program equivalence properties require the formalization of invariant properties that can be stated and proved on the prover side. However, in many cases, the user *knows* she/he is applying common techniques in such a way the generated proof obligations should be discharged without requiring any further justification on her/his side. Thus, this part implies as much automation as possible together with a clean specification of which invariant properties are preserved through compositionability of programing constructions. We shall also provide more involvement of already well established mathematical theories in the process. Martingale theory, strong laws and bounding expectation approximations should provide results as far as asymptotic properties are concerned.

Program equivalence

Observational equivalence provides a fundamental tool to reason about the behavior of programs. The objective is to develop a theory of observational equivalence for probabilistic programs, and to formalize a library of observational equivalence lemmas. In order to use these definitions and lemmas for game-based cryptographic proofs and computational non-interference proofs, we shall consider a notion of observational equivalence that is indexed by a relation I on input states and a relation O on output states, i.e. of the form $\simeq_O^I$; in many instances, the relations I and O are determined by sets of variables that characterize the initial and final knowledge of the attacker.

In addition, we shall study alternative notions of program equivalence based on indistinguishability, and formalize an equational theory for this notion of equivalence. For such a notion of equivalence, it will be important to take complexity issues into account : for example, some equivalence notions are only reflexive for polynomial-time programs.

Finally, some equivalence lemmas strongly rely on the linearity of program variables, i.e. on the fact that some variables are only written/read once. In order to state and prove these equivalences, we shall formalize a theory of variable usage ; furthermore, we shall implement certified decision procedures to ensure that linearity constraints are satisfied, in order to make the application of equivalence lemmas based on linearity policies as automated as possible.

Program transformation

The objective is to prove that program optimizations that are commonly performed by optimizing compilers are semantically correct, in the sense that the source and optimized programs are observationally equivalent. The primary motivation comes from WP3, and the fact that such program optimizations are commonly used in so-called bridging steps, i.e. steps that transform games into semantically equivalent ones.

One first objective is to implement and certify in Coq those program optimizations that are most commonly used in game-based proofs : constant propagation, common subexpression elimination, useless code elimination, function inlining, and code motion. There are opportunities for synergy with the ARA project CompCert on certified optimizing compilers, in which CPR-Cédric is involved (members of the EVEREST-INRIA team were also involved in the earlier project Concert).

A second, more theoretical objective, is to prove that every optimization that is semantics-preserving for deterministic programs remains semantics-preserving for probabilistic programs. In order to be able to formulate the problem precisely, it shall be necessary to resort to a generic framework to describe optimizations, along the lines of Rhodium [60]. Then, a possible proof strategy would consist in showing that the proof obligations that guarantee the soundness of the optimization in a deterministic setting are equivalent, or imply, the proof obligations that guarantee the soundness of the optimization in a probabilistic setting.

Milestones

Our approach in that package will be to design appropriate methods corresponding to the different classes of problems described above and to provide tools in order to mechanize these methods. These tools will consist of specialized libraries and specific tactics for partial automation (possibly using links to external tools for algebraic reasoning).

We plan the following milestones :

- D2.1 : T0 + 12 - first mathematical specifications of the needed invariance properties, program equivalence notions and program transformation for the languages defined in WP1.
- D2.2 : T0 + 18 - first version of Coq libraries for selected invariance properties, program equivalences and program transformations
- D2.3 : T0 + 24 - first prototype tools for mechanizing these methods, evaluation on case studies, decision of strategies for automation.
- D2.4 : T0 + 36 - final prototype tools and library

Rule	Description
Failure events	Maximize the difference in distributions between two programs that correspond until some failure events by the probability of the failure event
Motion of failure events	Establish an equivalence between failure events that arise in different parts of the program
Derandomization	Replace random assignments with assignments chosen by the attacker to maximize the probability of a failure event
Coin fixing	Delegate random assignments performed by the game but only used by the attacker to the latter

Fig. 1 – Some Game-Based Rules

WP3 Proofs of protocols, construction of cryptographic primitives

Leader : EVEREST-INRIA. Participants : DCS-VERIMAG. The objective of this workpackage is to provide an appropriate environment to verify formally game-based proofs of cryptographic systems, and to carry selected proofs within this environment. This workpackage builds upon the formalization of probabilistic programs developed in WP1, and the proof technology developed in WP2.

Environment

In order to be able to reason about game-based cryptographic proofs, the framework shall integrate the following components :

– *Specification of games* The first component of the tool extends the programming language developed in WP1 with a number of functionalities to perform a number of verifications on games, ranging from standard verifications (e.g. the distinction between variables and arrays, or the dimension of arrays are respected) to more specific verifications (e.g. the program respects the calling and variable policies imposed by the games).

– *Semantics of games* The second component of the tool is the programming language semantics, developed in WP1. Its role is two-fold : first, it is used to formalize security properties and security assumptions; second, it is used to justify the game-based proof techniques that are used in security proofs.

– *Security properties and assumptions* The third component of the tool is a library that defines standard cryptographic properties and assumptions. More concretely, we intend to develop code-based formalizations of properties and assumptions such as indistinguishability between distributions, IND-CCA for encryption primitives, CMA for signatures, DDH hypothesis, etc. Together with the first two components (and with the Coq type checker), it constitutes the generic trusted base of our tool.

– *Game-Based Proof rules* The fourth component provides users a collection of elementary rules to perform game-based security proofs. All rules are certified, in the sense that they are proved correct with respect to the operational semantics, or reduced to a security assumptions. The role of proof rules is to let users abstract away from the semantics of games; thus, the set or proof rules must be sufficiently complete so that users do not need to delve into the semantics of games to complete their proofs. The first objective is to provide automated support for the game-based rules that are most commonly used, including failure events, coin fixing, and derandomization (see Figure 1). Our objective here is to support sufficiently many game-based rules to perform the case studies described in the second paragraph. A second, more exploratory objective, is to develop additional proof rules that may be required by other case studies, related to blind computations or zero knowledge proofs.

Case studies

– *Reduction proofs between security criteria* As a sanity check of the correctness of our definitions and of the usability of our framework, we shall also machine-check the reduction proofs between different security properties, as given in [80]. A more ambitious objective could be to formalize the security criteria of Janvier, Lakhnech, Mazaré [53]. This would constitute an important step towards a proof of soundness of the formal model w.r.t. the computational model.
– *Proofs of cryptosystems* This is the main objective w.r.t. case studies. We intend to provide complete machine-checked game-based cryptographic proofs of increasing complexity. One initial goal is to formalize all the proofs of [78] ; doing so should be achievable within the first year of the project, since the examples are relatively simple (e.g. no arrays). A second objective is to machine-check the examples of [44], namely the tweakable enciphering scheme of Halevi-Rogaway, and the Cramer-Shoup cryptosystem, and the proof of OAEP of Bellare-Rogaway. These examples seem particularly appropriate since the proofs require all the techniques developed in WP2 and WP3. Note however that our objective is not only to consider the game-based arguments of the proof, that is coined as the mundane part of the proof by Halevi. We also intend to take advantage of the generality of the Coq proof system to carry the final steps of reasoning, that is coined as the creative part of the proof by Halevi, where a case analysis is performed to assert the security of the final game. In our opinion, performing the last part of the proof shall contribute convincing the cryptographers of the interest of our approach, and may also raise interesting opportunities for the development of tactics or libraries to reason about probabilities.
– *Applications to protocols* Password-based group key exchange protocols are a challenging application for out tool and present many interesting features. The first feature that distinguishes these protocols from the usual Clark and Jacob protocols [29] is that the number of participants in a session is not fixed. That is, participants can join and leave during session execution. The second feature is that passwords are short keys and are subject to guessing attacks. Our interest in these protocols stems from the fact that they present interesting and new challenges from the point of view of modeling and verification. (See `http://sky.fit.qut.edu.au/~choo/lounge.html` for a useful lounge of key exchange protocols.)

Relation to CryptoVerif

In addition to the above tasks, we shall pursue a more theoretical task, which does not (at least in a first stage) involve using proof assistant. The objective here is to make a precise comparison between our work and CryptoVerif, developed by Bruno Blanchet. Concretely, we intend to develop compilers from Blanchet's language to ours, and conversely, and study the class of properties for which these compilers are security-preserving. Potential benefits of these works are the certification of CryptoVerif, and its use as an external tool to generate sequences of games that can be certified automatically (each translation serves a different purpose).

Milestones

We plan the following milestones :
– D3.1 : T0 + 12 - machine-checked proofs of ElGamal and pseudo-random functions
– D3.2 : T0 + 24 - machine-checked proof of OAEP evaluation on case studies, decision of strategies for automation.
– D3.3 : T0 + 36 - a password-based group key exchange (GKE) protocol formalization
– D3.4 : T0 + 48 - proof of the GKE protocol

WP4 Computational flow of information

Leader : DCS-VERIMAG Participants : EVEREST-INRIA.

Non-interference is a high-level security property that guarantees the absence of illicit information leakages through executing programs. Non-interference for a program assumes a separation between secret variables and public variables and requires that executing the program in two initial states that coincide on the public variables leads to final states that coincide on the public variables. A common means to enforce non-interference is to use an information flow type system. In many applications, however, it is desirable to leak information in a controllable way. This is called declassification in the literature. In a useful survey, Sabelfeld & Sands [74] classify declassification techniques according to the following dimensions : "what", "who", "where" and "when". In this project, we are interested in the "when" dimension of declassification, and in particular in *cryptography-based declassification.* Our goal is to develop enforcement mechanisms that allow to leak secret sensitive data after it has been encrypted. For instance, we want that our enforcement mechanisms make it possible to assign to a public variable the encrypted value of a secret variable, provided the underlying encryption scheme is safe. More specifically, our aim is to develop **complexity-theoretically sound type systems for information flow and data integrity** for programs that use cryptographic primitives.

1. *Computational information flow for deterministic encryption.* Computational information flow is a very recent topic, even by the standards of language-based security, and existing results such as [58, 59, 9, 8] only deal with a few cryptographic primitives : one-way functions, and probabilistic symmetric encryption.

   The objective is to adapt existing results to further cryptographic primitives. As a first step, we intend to consider information flow type systems for a programming language that includes cryptographic primitives such as deterministic encryption based on pseudorandom permutations. Indeed, pseudorandom permutations are more "primitive" than the primitives previously considered, as they are often used to construct more sophisticated primitives satisfying stronger security properties ; see [76] for a detailed description of the modes of operation of block ciphers, and [21] for proofs of security. They are also easier to implement and less expensive, and in consequence, they are more suitable to be used in systems with limited computational or/and power capacities, as is the case for most of embedded systems.
2. *Computational information flow for password-based encryption* The goal is to adapt information flow type systems for a programming language that includes cryptographic password-based encryption. Most cryptographic primitives assume that secrets are uniformly distributed over a large space ; this is necessary to prove strong security guarantees, but at the same time requires additional cryptographic devices capable of storing high-entropy secret keys. Password-based key exchange protocols assume a more realistic scenario in which secret keys are chosen from a small known set of possible values, and this makes them human-memorable. The price to pay is that such weak secrets are often subject to so-called dictionary attacks : attacks in which an adversary tries to break the security of a scheme by trying all possible values of secret keys in a given small set (i.e., the dictionary). To address this problem, several protocols have been designed to be secure even when the secret key is a password. The goal of these protocols is to restrict the adversary's success to on-line guessing attacks [3, 24] .
3. *Computational information flow for integrity* The goal is to design type systems for data integrity and prove their complexity-theoretic soundness. An useful and critical application of cryptographic primitives and protocols is ensuring **integrity** : how to guarantee that sensitive data is not affected by a malicious adversary. Despite the fact that integrity is one of the most primitive security properties, at this moment no proved complexity-theoretic sound type system is known which deals with data integrity.
4. *Formalizing in type systems computational assumptions.* The aim is to develop a type system for non-interference for programs that use modular exponentiation. This necessitates to take into account computational assumptions such as Discrete Logarithm and Diffie-Hellman. The Discrete Logarithm assumption states that it is difficult to compute x given g^x. The computational DH assumption states that given g, g^x and g^y, it is difficult to compute g^{xy}. There is also a decisional version. Taking into account such computational assumptions is necessary to prove the correctness of many cryptographic constructions and is a new and non-trivial challenge [15].

Milestones

We plan the following milestones :
- D4.1 : T0 + 12 - Type system for deterministic and password-based encryption
- D3.2 : T0 + 24 - machine-checked proof of the computational soundness of the type system of D4.1
- D3.3 : T0 + 36 - Type system for a language with modular exponentiation
- D3.4 : T0 + 48 - machine-checked proof of the computational soundness of the type system of D4.3

WP5 Watermarking algorithms

Leader : CPR-Cédric. Participants : DCS-VERIMAG.

This task is dedicated to studying the specific security concern of certifying *watermarking* algorithms and provide a framework to certify such algorithms. It relies on the formalization of probabilistic programs developed in WP1 and on game based security proofs of WP3. To achieve this goal we need to perform the following tasks :

Environment

- *modelling attacks* A significant part of the work is to model different kinds of attacks. A classical attempt to remove a mark consists in random alteration of data. There are other kinds of attack depending on the kind of data. On semi-structured data [6] for instance, random deletions of n-uples and random insertion of unmarked n-uples are well known attacks.
  Note that these attacks are in essence probabilistic : we shall model them as probabilistic programs. Hence, the robustness property itself is probabilistic : given a kind of attack, the probability of success of the attack (i.e. removing the mark without destroying the data) must be low.
- *modelling pseudo-random generators* is a necessary step toward a complete certification.
- *modelling watermarking properties* using game-based proofs of WP3. It is known that weak watermarking algorithm allow an attacker to retrieve information on a key k from several data marked with k. This should be formalized and proved by means of game-based semantics.

Case studies

- *Proof of Watermarking algorithms* We focus on certifying a recent watermarking algorithm by Agrawal, Kiernan and Haas [5], introduced in 2003 and tackling the problem of watermarking databases. An extension of this algorithm [54] by Gross-Amblard and Lafaye is implemented in the watermarking system `Watermill` [30]. Firstly we shall model the AKH algorithm and prove some properties including the robustness property. Secondly we shall address the extension to quality-preserving requests in [54]. This may open a way towards a verification of its implementation, using the techniques and tools developed in this project.
  This study shall emphasize the problems arising from poorly specified/implemented pseudo-random generators.

Milestones

- D5.1 : T0 + 12 - Develop and prove a realistic algorithm example using the language of WP1.
- D5.2L T0 + 24 - Develop a framework dedicated to certification of watermarking. It should be based on program equivalence of WP2 and may also use the notions developed in WP3 and WP4.
- D5.3 : T0 + 48 - complete framework, with random generator properties. Proofs of attacks exploiting bad (use of) generators.

Liste des livrables / List of deliverable

Remark In the following table, under the column Nature, R stands for Report and S for Software (including formal proofs). Under the column $T_0 + \ldots$, the date of delivery is given as the number of elapsed months from the beginning of the project. We plan a yearly release for the software deliverables and a workshop at the end of the project.

Nb	Title	Nature	Resp.	Participants	$T_0 + \ldots$
D0.1	SCALP website	S	DCS	all	3
D0.2	Activity report	R	EVEREST	all	6
D0.3	Activity report	R	ProVal	all	12
D0.4	Activity report	R	Plume	all	18
D0.5	Activity report	R	DCS	all	24
D0.6	Activity report	R	CPR	all	30
D0.7	Activity report	R	EVEREST	all	36
D0.8	Activity report	R	ProVal	all	42
D0.9	Final activity report	R	DCS	all	48
D1.1	Simple core language	R	Plume	all	6
D1.2	Formalisation of the simple core language in Coq	S	Plume	all	12
D1.3	Core language assessment	R	Plume	all	24
D1.4	Core language and its formalization	R-S	Plume	all	36
D2.1	Mathematical specifications of selected properties	S	ProVal	all	12
D2.2	Coq libraries for selected properties	S	ProVal	all	18
D2.3	Prototype tools for mechanizing proofs	S	ProVal	all	24
D2.4	Final prototype tools and library	S	ProVal	all	36
D3.1	ElGamal and pseudo-random functions	S	EVEREST	EVEREST,CPR	12
D3.2	OAEP, automation strategies	S	EVEREST	EVEREST	24
D3.3	Formalization of a password-based GKE protocols	S	EVEREST	EVEREST,DCS	36
D3.4	Proof of the GKEP protocol	S	EVEREST	EVEREST,DCS	48
D4.1	Type system for ciphers and password	R	DCS	DCS,EVEREST	12
D4.2	Machine-checked proof of type system of D4.1	S	DCS	DCS,EVEREST	24
D4.3	Modular exponentiation	R	DCS	DCS,EVEREST	36
D4.4	Machine-checked proof of type system of D4.3	S	DCS	DCS,EVEREST	48
D5.1	Case study	S	CPR	CPR	12
D5.2	Certification framework of watermarking	S	CPR	CPR,DCS	24
D5.3	Complete framework, with random generator properties	S	CPR	CPR,DCS	48

Résultats escomptés – perspectives / Expected results and perspectives

The project will deliver a set of tools for proving the correctness of cryptographic systems, and shall demonstrate its effectiveness on three application domains : cryptographic primitives and protocols, type systems for computational information flow, and watermarking algorithms. Therefore the success of the project will measured by standard means, such as publications in international conferences and journals, as well as by the development of formalized libraries and by their application to state-of-the-art case studies. Being the culminating point of the research project, the complete formalization of case studies is the most ambitious criterion for success. We detail the criterion for each case study below :

- *Cryptographic primitives and protocols* The criterion of success w.r.t. WP3 is a complete formalization of advanced uses of the game-playing techniques for proving the correctness of both cryptographic schemes and protocols. In order to maximize impact, our main priority is to certify at least the proof of OAEP, and of a password-based group protocol.
- *Computationally sound information flow* The criterion of success w.r.t. WP4 is the complete machine-checked definition and metatheoretical study of type-systems supporting at least two of the four following features : deterministic encryption, password-based encryption, integrity checks, computational assumptions.
- *Watermarking case study* The criterion of success w.r.t. the watermarking case study is a complete prototype environment for certifying watermarking algorithms. The ultimate goal of this case study is to tackle difficult robustness notions on *constrained semi-structured data* [30, 35].

The project members are aware that the project is ambitious and measure the risks. We think, however, that the development of a prototype capable of handling substantial case studies is a reasonable goal. Probably, the main risk is that the case studies are not fully finalized by the end of the project. However, even partial success shall be valuable, and will pave the way for further research towards the objective. Furthermore, we have already identified intermediate results that are interesting on their own, including :

- A probabilistic language for cryptosystems and cryptoproofs with formalized semantics and reasoning.
- A library of proved program transformations based on code optimization techniques.
- Rigorous definitions and reasoning of observational equivalence relations for probabilistic programs.
- A library of proofs of well-known properties about measures of classical distributions (Binomial,...).
- A framework for proving probabilistic watermarking algorithms, based on libraries above.
- A library of lemmata and tactics that support reductionist proofs.
- A Coq-proved type system for non-interference and data integrity for a programming language that includes random number generation, block ciphers and password-based encryption. The correctness we seek is complexity-theoretic.
- A library of results about random generators.

Impact

Although it is not realistic to expect that the project will conclude with the adoption of formal proofs in the application domains it spans over, we shall thrive to promote the use of formal proofs in these communities by publishing in relevant security conferences, as well as in conferences in formal methods, programming languages, and verification. One long term objective of the consortium members is to push the cryptography community to perform machine-checked proofs of cryptographic schemes as a means to overcome existing difficulties with pencil-and-paper proofs, in the same way that the POPLmark challenge has pushed the programming language community to perform machine-checked proofs of their work. A related objective is to promote the use of our tools and libraries among "formal" cryptographers ; in particular, we hope that the formal developments produced by the project will be used to machine-check proofs of soundness of the symbolic model, and in a second phase to develop certified decision procedures for the computational model, using a combination of reflective techniques based on decidability results for the symbolic model and on the aforementioned soundness results.

As we mentioned earlier, there are few tools for quantitative analysis of randomized programs. This project will contribute to establish a well-developed toolbox on top of the Coq proof assistant for modeling, specifying and analyzing a large class on randomized algorithms, not only the programs involved in security protocols.

Propriété intellectuelle / Intellectual property

The inputs of this project are scientific knowledge and software tools. This scientific knowledge is publicly available, and these software tools all qualify as free/open-source software under the definition of the Free Software Foundation, the Debian Free Software Guidelines, and the Open Source Initiative.

The outputs will be publicly disseminated as research reports and articles in conferences and journals. Our software tools will be distributed under the “CEA CNRS INRIA logiciel libre” licence (CeCILL), which is compatible with the GNU GPL licence.

Moyens financiers demandés / Financial resources

DCS-VERIMAG

- Permanent researchers supplied by the project : 4 man-years, for a total cost of 315550€
- Non-permanent researchers requested for the project : 4 man-years for a total cost of 135711€. Decomposed as follows :
  - Thesis on *Computationally sound type systems for information flow* : 92052€ (3 years at 30684€ per year). See page 29 for a complete description.
  - Shared post-doc with Plume-LIP : 43659€ (1 year for each partner)
- Funds requested for small equipment : 5276€. Corresponding to three laptops (3 × 1500€ plus the 17.25% "TVA non récupérable" tax).
- Funds requested for missions expenses : 14070€. That is, an average of 1500€ per man-year involved in the project (8 man years), for attending an international conference and two project meetings, plus the 17.25% "TVA non récupérable" tax.

EVEREST-INRIA

- Permanent researchers supplied by the project : 3.8 man-years, for a total cost of 198192€.
- Non-permanent researchers requested for the project : a 3-year PhD Thesis on "Certification of cryptographic algorithms and protocols " for 101010€.
- Funds requested for small equipment : 9849€. Corresponding to 1 workstation (2.5k€ HT), 2 laptops (2 × 3k€ HT) + the "TVA non récupérable" tax (15.87%).
- Funds requested for missions expenses : 15000€ (4 project reunion per year for 2 plus 1 international conference per year for one).

Plume-LIP

- Permanent researchers supplied by the project : 1.5 man-years, for a total cost of 80229€
- Non-permanent researchers requested for the project : 1.5 man-years for a total cost of 58800€ decomposed as follows :
  - Shared post-doc with DCS-VERIMAG : 43659€ (1 year for each partner)
- Funds requested for small equipment : 1876€.
- Funds requested for missions expenses : 11725€. (2 attendances to an international conference per year plus attendance to project meetings for a total of 2500€ per year plus the 17.25% "TVA non récupérable" tax.)

ProVal-LRI

- Permanent researchers supplied by the project : 1 man-year. A post-doc funded by INRIA has also been asked in 2007 for working on fundamental aspects related to this project.
- Non-permanent researchers requested for the project : 2 man-years for a total cost of 105840€. Decomposed as follows :
  - A post-doc (1 year) to be hired in 2008 who will contribute to the general platform for analyzing probabilistic programs, in particular in the domain of proof automation.
  - 2 or 3 interns starting in 2008 working for 4 to 6 months (total 1 year). They will be asked to develop libraries and examples in order to improve the platform.
- Funds requested for small equipment : 4100€. Corresponding to two powerful workstations and 1 laptop.
- Funds requested for missions expenses : 16000€. The outcome of the project will be publications in international conferences in the area of security and proof assistants. We plan to participate to these conferences and give opportunities for young researchers to attend summerschools in these areas.

CPR-Cédric

- Permanent researchers supplied by the project : 1.4 man-year, for a total cost of 74880€.

- Non-permanent researchers requested for the project : none.
- Funds requested for small equipment : 13000€. Corresponding to one powerful station needed for developing proofs of realistic programs, and two laptops.
- Funds requested for missions expenses : 22000€. intra-project meetings plus 1 international conference per year for 2 plus invitation of external researchers to give talks plus frequent visits to D. Gross-Amblard in Dijon.

Experts / Experts

Suggestion d'expert pour l'évaluation				
Prénom	Nom	Courriel	Affiliation (labo/entreprise/..)	Domaine(s) d'expertise
David	Lubicz	`david.lubicz @univ-rennes1.fr`	Centre d'ELectronique de l'ARmement	Cryptography
Joshua	Guttman	`guttman@mitre.org`	The MITRE Corporation	Formal verification of cryptographic protocols
Olivier	Bournez	`Olivier.Bournez@loria.fr`	INRIA-Lorraine	probabilistic reasoning
Joe	Hurd	`joe.hurd@magd.ox.ac.uk`	Oxford Computing Lab.	Theorem proving, Applications to cryptography
Bruno	Blanchet	`Bruno.Blanchet@ens.fr`	ENS	Formal and computational verification

CV of project members

Lakhnech Yassine. [participation 50%]
Né le 8 février 1964

Cursus/Carrière :
Actuellement : Professeur à Verimag
1996-1999 : Professeur associé à la Christian-Albrechts Universität zu Kiel, Allemagne.

Domaine d'expertise : sémantique et vérification des programmes, sémantique et vérification des protocoles cryptographiques.

Publications : Plus de 44 publications internationales dont 8 revues.
1 - L. Bozga, C. Ene and Y. Lakhnech. A symbolic decision procedure for cryptographic protocols with time stamps. *Journal of Logic and Algebraic Programming*, 65(1), pages 1–35, 2005.
2 - L. Bozga, Y. Lakhnech and M. Périn. Pattern-based abstraction for verifying secrecy in protocols. *International Journal on Software Tools for Technology Transfer* 8(1) : 57-76, 2006. 3 - M. Bozga, R. Iosif, Y. Lakhnech : Flat Parametric Counter Automata. In *Proc. of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*, pages 577–588, Venice, Italy, 2006.

Ene Cristian. [participation 30%]
Né le 03 octobre 1971

Cursus/Carrière :
Actuellement : Maître de Conférence, Verimag, Université Joseph Fourier, Grenoble
2002-2004 : Post-doctorant, Verimag, Grenoble

Domaine d'expertise : modèles pour la concurrence et la mobilité, protocoles de sécurité, approche formelle et computationelle

Publications : Plus de 20 publications, dont 7 revues internationales et 7 colloques internationaux.
1 - L. Bozga, C. Ene, R. Janvier, Y. Lakhnech, L. Mazaré and M. Périn. Automatic Verification of Security Properties Based on Abstractions. In *Proc. of Verification of Infinite-State Systems with Applications to Security (VISSAS'05)*, pages 23–53, 2005.
2 - L. Bozga, C. Ene and Y. Lakhnech. A symbolic decision procedure for cryptographic protocols with time stamps. *Journal of Logic and Algebraic Programming*, 65(1) : 1–35, 2005.
3 - L. Bozga, C. Ene, Y. Lakhnech. On the Existence of an Effective and Complete Inference System for Cryptographic Protocols. In *Proc. of the 7th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'04)*, LNCS 2987, pages 42–57, Barcelona, Spain, 2004.

COURANT Judicaël. [participation 50%]
Né le 20 février 1973

Cursus/Carrière :
Actuellement : Collaborateur expert du laboratoire Verimag, Équipe DCS
2004-2005 : Consultant indépendant informatique puis ingénieur chez Atos Worldline
2003-2004 : Année sabbatique (engagement associatif)
1998-2003 : Maître de Conférence, Université Paris XI, LRI.

Domaine d'expertise : méthodes formelles, vérification assistée, théorie des types, approche formelle et computationelle.

Publications : Une vingtaine de publications, dont une revue internationale et 6 conférences internationales avec comité de lecture. Participation au développement de l'assistant à la démonstration Coq de 1994 à 2003.
1 - J. Courant. $\mathcal{MC}_2$: A Module Calculus for Pure Type Systems. *Journal of Functional Programming.* 66 pages. To appear.
2 - J. Courant and J.-F. Monin. Defending the bank with a proof assistant. In *Proc. of the 6th International IFIP WG 1.7 Workshop on Issues in The Theory of Security (WITS'06)*, pages 87–98, Vienna, Austria, 2006.
3 - J. Courant. Explicit universes for the calculus of constructions. In *Proc. of the 15th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'02)* , volume 2410, pages 115–130, Hampton, VA, USA, 2002.

MONIN Jean-François. [participation 20%]
Né le 24 juin 1960

Cursus/Carrière :
Actuellement : Professeur à l'université Joseph Fourier, Grenoble 1, laboratoire Verimag
1983-2003 : Ingénieur (y compris responsabilité d'équipe) au Centre National d'Etudes des Télecommunications (CNET) puis France Télécom R&D, Lannion.

Domaine d'expertise : méthode formelles, vérification assistée, API de sécurité.

Publications : Plus de 30 publications, principalement dans des colloques internationaux ainsi qu'un ouvrage traduit en anglais.
1 - J. Courant and J.-F. Monin. Defending the bank with a proof assistant. In *Proc. of the 6th International IFIP WG 1.7 Workshop on Issues in The Theory of Security (WITS'06)*, pages 87–98, Vienna, Austria, 2006.
2 - J.-F. Monin and J. Courant. Proving termination using dependent types : the case of xor-terms. In *Proc. of Trends in Fu nctional Programming (TFP'06)*, Nottingham, 2006.
3 - J.-F. Monin and Ph. Chavin. Coq. In H. Habrias and M. Frappier, editors, *Software Specification Methods, An Overview Using a Case Study*, ISTE, chapter 16. Hermès Science, April 2006.

BARTHE Gilles. [participation 25%]
Né le 1 Octobre 1967

Cursus/Carrière :
Actuellement : Directeur de recherche à l'INRIA Sophia-Antipolis.

Domaine d'expertise : sécurité du logiciel, assistant de preuves (Coq), méthode formelles, vérification formelle de primitives cryptographiques.

Publications : Plus de 60 publications, dont 3 ouvrages édités.
1 - Gilles Barthe, Jan Cederquist, Sabrina Tarento : A Machine-Checked Formalization of the Generic Model and the Random Oracle Model. In D. Basin and M. Rusinowitch, editors, Proceedings of IJCAR 2004, volume 3097 of Lecture Notes in Computer Science, pages 385-399, Cork, Ireland, July 2004. Springer-Verlag
2 - Gilles Barthe, David Naumann, and Tamra Rezk. Deriving an information flow checker and certifying

compiler for Java. In Proceedings of Symposium of Security and Privacy 2006, Oakland, May 2006. IEEE Press
3 - G. Barthe, P. Courtieu, G. Dufay, and S. Melo de Sousa. Jakarta : tool-assisted specification and verification of the JavaCard Platform. Journal of Automated Reasoning, 2006.

Grégoire Benjamin. [participation 40%]
Né le 20 Septembre 1975.

Cursus/Carrière :
Actuellement : Chargé de recherche à l'INRIA Sophia-Antipolis.

Domaine d'expertise : assistant de preuves (Coq), logique d'ordre supérieur, vérification assistée.

Publications : plus de 10 publications, dont 1 ouvrage édité.
1 - B. Grégoire, L. Théry, and B. Werner. A computational approach to Pocklington certificates in type theory. In M. Hagiya and P. Wadler, editors, Proceedings of FLOPS'06, volume 3945 of Lecture Notes in Computer Science, pages 97 - 113. Springer-Verlag, 2006.
2 - B. Grégoire and L. Théry. A purely functional library for modular arithmetic and its application for certifying large prime numbers. In U. Furbach and N. Shankar, editors, Proceedings of IJCAR'06, volume 4130 of Lecture Notes in Artificial Intelligence, pages 423-437. Springer-Verlag, 2006.
3 - B. Grégoire and A. Mahboubi. Proving equalities in a commutative ring done right in Coq. In J. Hurd and T. Melham, editors, Proceedings of TPHOLs'05, volume 3603 of Lecture Notes in Computer Science, pages 98-113, Oxford, UK, August 2005. Springer-Verlag.

Zanella Béguelin Santiago. [participation 30%]
Né le 8 Septembre 1981.

Cursus/Carrière :
Actuellement : Doctorant à l'INRIA Sophia-Antipolis depuis Juin 2006.

Domaine d'expertise : méthode formelles, vérification assistée, sécurité du logiciel.

Publications : 2 publications.
1 - S. Zanella Béguelin, G. Betarte, and C. Luna. A formal specification of the MIDP 2.0 security model. In Proc. 4th International Workshop on Formal Aspects in Security and Trust, FAST 2006, Hamilton, Canada, August 26-27 2006, Lectures Notes in Computer Science. Springer 2006.
2 - S. Zanella Béguelin. Formalisation and verification of the GlobalPlatform Card Specification using the B method. In G. Barthe, B. Grégoire, M. Huisman, and J.-L. Lanet, editors, Construction and Analysis of Safe, Secure, and Interoperable Smart Devices, Second International Workshop, CASSIS 2005, Nice, France, March 8-11 2005, volume 3956 of Lectures Notes in Computer Science, pages 155-273. Springer, 2006.

Paulin-Mohring Christine. [participation 25%]
Née le 31 mai 1962

Cursus/Carrière :
Actuellement : Professeur à l'université Paris-Sud, détachée comme directeur de recherche à l'INRIA Futurs en 2006-2007 (renouvellement demandé pour 2007-2008).

Domaine d'expertise : assistant de preuves (Coq), logique d'ordre supérieur, méthode formelles, vérification assistée.

Publications : Plus de 30 publications, dont 11 conférences internationales, 2 revues et 3 ouvrages édités. Coordination du développement de l'assistant de preuves Coq de 1994 à 2004.
1 - Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. In Tarmo Uustalu, editor, Mathematics of Program Construction, MPC 2006, volume 4014 of Lecture Notes in Computer Science, Kuressaare, Estonia, July 2006. Springer-Verlag
2 - Claude Marché, Christine Paulin-Mohring, and Xavier Urbain. The Krakatoa tool for certification of Java/JavaCard programs annotated in JML. Journal of Logic and Algebraic Programming, 58(1-2) :89-106, 2004
3 - June Andronick, Boutheina Chetali, and Christine Paulin-Mohring. Formal verification of security properties of smart card embedded source code. In John Fitzgerald, Ian J. Hayes, and Andrzej Tarlecki, editors, International Symposium of Formal Methods Europe (FM'05), volume 3582 of Lecture Notes in Computer Science, Newcastle,UK, July 2005. Springer-Verlag

AUDEBAUD Philippe. [participation 25%]
Né le 14 septembre 1956

Cursus/Carrière :
Actuellement : Maître de Conférences à l'École Normale Supérieure de Lyon. Délégation de 2001 à 2006 à l'INRIA Sophia-Antipolis et l'Université de Nice Sophia-Antipolis.

Domaine d'expertise : sémantique des langages de programmation, modèles du lambda-calcul, logique d'ordre supérieur, interfaces assistants de preuves.

Publications : 2 revues et 3 conférences internationales.
1 - Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. In Tarmo Uustalu, editor, Mathematics of Program Construction, MPC 2006, volume 4014 of Lecture Notes in Computer Science, Kuressaare, Estonia, July 2006. Springer-Verlag.
2 - Philippe Audebaud and Laurence Rideau. TeXmacs as authoring tool for publication and dissemination of formal developments. Electronic Notes in Theoretical Computer Science, 103, 2004.
3 - Philippe Audebaud and Elena Zucca. Deriving proof rules from continuation semantics. Formal Aspects of Computing, 11(4) :426-447, 1999.

COURTIEU Pierre. [participation 20%]
Né le 14 juillet 1973

Cursus/Carrière :
Actuellement : Maître de Conférences au Conservatoire National des Arts et Métiers (Paris).

Domaine d'expertise : Assistants de preuves. λ-calculs. Preuves automatiques.

Publications : 1 revues et 6 conférences internationales.
1- G. Barthe and P. Courtieu. Efficient Reasoning about Executable Specifications in Coq. In C. Mu oz V. A Carreno and S. Tahar, editors, Proceedings of TPHOLs'02, volume 2410 of Lecture Notes in Computer Science, pages 31-46. Springer, 2002.
2- G. Barthe, P. Courtieu, G. Dufay, and S. Melo de Sousa. Jakarta : tool-assisted specification and verification of the JavaCard Platform. Journal of Automated Reasoning, 2006. To appear.
3- Pierre Courtieu. Normalized types. In Proceedings of CSL2001, volume 2142 of Lecture Notes in Computer Science, 2001.

URBAIN Xavier. [participation 15%]
Né le 7 juin 1973

Cursus/Carrière :
Actuellement : Maître de Conférences à l'École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE).
Domaine d'expertise : Preuves automatiques et certification. Récriture. Terminaison. Vérification de code.

Publications : 5 revues internationales et 3 conférences internationales.
1- Evelyne Contejean, Claude Marché, Ana Paula Tomás et Xavier Urbain. Mechanically proving termination using polynomial interpretations. Journal of Automated Reasoning, 34(4) pp 325–363, 2005.
2- Xavier Urbain. Modular & Incremental Automated Termination Proofs. Journal of Automated Reasoning, 32(4) pp 315–355, 2004.
3- Claude Marché, Christine Paulin et Xavier Urbain. The Krakatoa Tool for JML/Java Program Certification. Journal of Logic and Algebraic Programming, 58(1–2) pp 89–106, 2004.

PhD Thesis Subjects

Computationally sound type systems for information flow

Supervisor : Yassine Lakhnech and Judicaël Courant, at DCS-VERIMAG.

Type systems for secure information flow aim to prevent a program from leaking information from variables that hold secret data to variables that hold public data. Such system have already been provided over past years, based on sound semantics w.r.t. the symbolic model of cryptography. However, these systems have reached their limits when trying to consider random values, or deterministic encryption : providing correct symbolic model is sound in these cases (for instance, encrypting a secret should not yield a public value for deterministic encryption). Therefore, computationally sound type systems have to be provided. Still, proving correctness of these types systems remains difficult.

The objective of this thesis is :

- To perform a Coq-checked proof of some already hand-checked proof of computational soundness of a type system for information flow.
- To provide theoretical tools in order to assist writing these kind of proofs.
- To implement these tools in Coq or around Coq.

Certification of cryptographic algorithms and protocols

Supervisor : Gilles Barthe, at EVEREST-INRIA.

No matter how carefully crafted cryptographic systems are, experience has shown that effective attacks can remain hidden for years. Thus, cryptographers increasingly advocate provable security, where new systems are published with a rigorous definition of their security goals and a mathematical proof that they meet their goals. While the adoption of provable security significantly enhances confidence in cryptosystems, the increasing complexity and diversity of the and diversity tends to increase, the community has become aware that the point has been reached where it is no longer viable to construct or verify cryptographic proofs by hand. Shoup [78] and Bellare and Rogaway [23], and Halevi [44] advocate the construction of cryptographic proofs as sequences of probabilistic games as a natural solution for taming the complexity of the task. The basic idea of these works is to use a fully-specified programming language to code those games and to use language-based techniques (observational equivalence, program transformations) for manipulating and reasoning about them.

The objective of the thesis is to provide the necessary infrastructure to formalize game-based proofs, and to use the infrastructure to prove the correctness of cryptographic schemes and information flow type systems for languages with cryptographic primitives. During the first year of the thesis, the objective is to contribute to the formalization of a probabilistic programming language of games, to develop the machinery required for game-based proofs (e.g. reflective tactics to check that read and write policies are correctly enforced), and to formalize a library of security properties expressed in a game-based setting. The objective of the second year is to develop theories of observational equivalences, and to develop tactics for the rules described in [22] (thus dealing with program optimizations, as well as specific rules for game-based proofs). In order to validate the applicability of the rules, the student shall simultaneously perform small-size case studies. At the end of the second year, a formalization of the main steps of OAEP should be complete. The objective of the third year is to apply these results to group protocols.

We also expect the student to be marginally involved in machine checking proofs of computational soundness of information flow type systems for languages with cryptographic primitives, since many of the libraries that will be developed by the student will also be useful for this purpose.

At the end of the thesis, the student will have a double expertise in cryptography and machine-checked proofs using the Coq proof assistant, and have published in conferences in cryptography, programming languages, and verification.

Références

[1] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Theoretical Aspects of Computer Software, 4th International Symposium*, volume 2215 of *Lecture Notes in Computer Science*, pages 82–94. Springer, 2001.

[2] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.

[3] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *Public Key Cryptography*, pages 65–84, 2005.

[4] Gul Agha, Jose Meseguer, and Koushik Sen. Pmaude : Rewrite-based specification language for probabilistic object systems. *ENTCS*, 153(2) :213–239, 2006. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[5] Rakesh Agrawal, Peter J. Haas, and Jerry Kiernan. A system for watermarking relational databases. In Alon Y. Halevy, Zachary G. Ives, and AnHai Doan, editors, *SIGMOD Conference*, page 674. ACM, 2003.

[6] Rakesh Agrawal and Jerry Kiernan. Watermarking Relational Databases. In *VLDB*, 2002.

[7] Stefan Katzenbeisser Andr Adelsbach and Ahmad-Reza Sadeghi. A computational model for watermark robustness. In *Proceedings of 8th Information Hiding Workshop 2006*, 2003.

[8] Aslan Askarov, Daniel Hedin, and Andrei Sabelfeld. Cryptographically-masked flows. In Kwangkeun Yi, editor, *SAS*, volume 4134 of *Lecture Notes in Computer Science*, pages 353–369. Springer, 2006.

[9] Aslan Askarov and Andrei Sabelfeld. Security-typed languages for implementation of cryptographic protocols : A case study. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 197–221. Springer, 2005.

[10] Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. In Tarmo Uustalu, editor, *Mathematics of Program Construction, MPC'2006*, volume 4014 of *LNCS*, Kuressaare, Estonia, 2006. Springer-Verlag.

[11] Philippe Audebaud and Christine Paulin-Mohring. Reasoning on probabilistic functional programs. Manuscript, 2007.

[12] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of the Tenth ACM conference on Computer and Communication Security*, pages 220–230. ACM Press, 2003.

[13] M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In Springer-Verlag, editor, *8th European Symposium on Research in Computer Security (ESORICS 2003)*, volume 2808 of *LNCS*, 2003.

[14] M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. In Springer-Verlag, editor, *1st Theory of Cryptography Conference (TCC)*, volume 2951 of *LNCS*, pages 271–290, 2004.

[15] Gergei Bana, Payman Mohassel, and Till Stegers. Computational soundness of formal indistinguishability and static equivalence. In *ASIAN 06*, LNCS. Springer-Verlag, 2006.

[16] G. Barthe, J. Cederquist, and S. Tarento. A Machine-Checked Formalization of the Generic Model and the Random Oracle Model. In D. Basin and M. Rusinowitch, editors, *Proceedings of IJCAR'04*, volume 3097 of *LNCS*, pages 385–399, 2004.

[17] Gilles Barthe, Pierre Courtieu, Guillaume Dufay, and Simão Melo de Sousa. Tool-assisted specification and verification of typed low-level languages. *J. Autom. Reasoning*, 35(4) :295–354, 2005.

[18] Gilles Barthe and Sabrina Tarento. A machine-checked formalization of the random oracle model. In Jean-Christophe Filliâtre, Christine Paulin-Mohring, and Benjamin Werner, editors, *Proceedings of TYPES'04*, volume 3839 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2004.

[19] M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. PhD thesis, LSV, ENS Cachan, 2007.

[20] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Proc. CRYPTO 94*, volume 839 of *LNCS*, pages 341–358, 1994.

[21] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403, 1997.

[22] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`.

[23] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[24] Steven M. Bellovin and Michael Merritt. Encrypted key exchange : Password-based protocols secure against dictionary attacks. In *IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, 1992.

[25] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *S&P*, pages 140–154. IEEE Computer Society, 2006.

[26] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 2006.

[27] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols (the case of encryption-based mutual authentication and key exchange). Cryptology ePrint Archive, Report 2004/334, 2004.

[28] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Errors in computational complexity proofs for protocols. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 624–643. Springer, 2005.

[29] John A. Clark and Jeremy L. Jacob. A survey of authentication protocol literature. Technical Report 1.0, 1997.

[30] Camelia Constantin, David Gross-Amblard, Meryem Guerrouani, and Julien Lafaye. Logiciel Watermill. `http://cedric.cnam.fr/vertigo/tadorne/soft/soft.html`.

[31] Evelyne Contejean, Claude Marché, Benjamin Monate, and Xavier Urbain. Proving termination of rewriting with CiME. In Albert Rubio, editor, *Extended Abstracts of the 6th International Workshop on Termination, WST'03*, pages 71–73, June 2003. `http://cime.lri.fr`.

[32] Evelyne Contejean, Claude Marché, Ana Paula Tomás, and Xavier Urbain. Mechanically proving termination using polynomial interpretations. *J. of Automated Reasoning*, 34(4) :325–363, 2005.

[33] Ricardo Corin and Jerry den Hartog. A probabilistic hoare-style logic for game-based cryptographic proofs. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2006.

[34] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In Sagiv [75], pages 157–171.

[35] Patrick Cousot and Radhia Cousot. An abstract interpretation-based framework for software watermarking. In *Principles of Programming Languages 2003*, 2003.

[36] Ingemar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom. *Digital Watermarking.* Morgan Kaufmann Publishers, Inc., San Francisco, 2001.

[37] Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *CSFW*, pages 321–334, 2006.

[38] S. Delaune. *Vérification des protocoles cryptographiques et propriétés algébriques.* PhD thesis, LSV, ENS Cachan, 2006.

[39] Alexander W. Dent. Fundamental problems in provable security and cryptography. *Phil Trans R Soc A*, 364 :3215–3230, 2006.

[40] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2) :198–208, 1983.

[41] Jean-Christophe Filliâtre. The why verification tool, 2002. http ://why.lri.fr/.

[42] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, April 1984.

[43] Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors. *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, volume 4189 of *Lecture Notes in Computer Science*. Springer, 2006.

[44] Shai Halevi. A plausible approach to computer-aided cryptographic proofs. `http://theory.lcs.mit.edu/~shaih/pubs.html`, 2005.

[45] Jounaidi Ben Hassen and Sofiène Tahar. On the numerical verification of probabilistic rewriting systems. In *DATE '06 : Proceedings of the conference on Design, automation and test in Europe*, pages 1223–1224. European Design and Automation Association, 2006.

[46] T. Herault, R. Lassaigne, and S. Peyronnet. Apmc 3.0 : Approximate verification of discrete and continuous time markov chains. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST 2006)*. IEEE, 2006.

[47] J. Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, June 2005.

[48] Joe Hurd. A formal approach to probabilistic termination. In Victor A. Carreño, César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics*, volume 2410 of *LNCS*, pages 230–245, Hampton, VA, USA, 2002. Springer-Verlag.

[49] Joe Hurd. *Formal Verification of Probabilistic Algorithms*. PhD thesis, Univ. of Cambridge, 2002.

[50] Joe Hurd. Verification of the Miller-Rabin probabilistic primality test. *J. of Logic and Algebraic Programming*, 50(1–2) :3–21, 2003. Special issue on Probabilistic Techniques for the Design and Analysis of Systems.

[51] Joe Hurd, Annabelle McIver, and Carroll Morgan. Probabilistic guarded commands mechanized in HOL. In A. Cerone and A. Di Pierro, editors, *Quantitative Aspects of Programming Languages*, volume 112 of *ENTCS*, pages 95–111, Barcelona, Spain, 2005. Elsevier.

[52] R. Janvier. *Liens entre modeles symboliques et computationnels pour les protocoles cryptographiques utilisant des hachages*. PhD thesis, Universtié de Grenoble 1, 2006.

[53] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Completing the picture : Soundness of formal encryption in the presence of active adversaries. In Sagiv [75], pages 172–185.

[54] Julien Lafaye and David Gross-Amblard. XML Streams Watermarking. In *20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec2006), Sophia An*, 2005.

[55] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with PRISM : A hybrid approach. *J. on Software Tools for Technology Transfer*, 2004.

[56] Pascal Lafourcade. *Vérification des protocoles cryptographiques en présence de théories éequationnelles*. PhD thesis, LSV, ENS Cachan, 2006.

[57] P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *IEEE Symposium on Security and Privacy*, pages 71–85, 2004.

[58] Peeter Laud. Secrecy types for a simulatable cryptographic library. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 26–35. ACM, 2005.

[59] Peeter Laud and Varmo Vene. A type system for computationally secure information flow. In Maciej Liskiewicz and Rüdiger Reischuk, editors, *FCT*, volume 3623 of *Lecture Notes in Computer Science*, pages 365–377. Springer, 2005.

[60] Sorin Lerner, Todd Millstein, Erika Rice, and Craig Chambers. Automated soundness proofs for dataflow analyses and transformations via local rules. In *Conference Record of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2005)*, Long Beach, California, January 2005.

[61] Claude Marché, Christine Paulin-Mohring, and Xavier Urbain. The KRAKATOA tool for certification of JAVA/JAVACARD programs annotated in JML. *J. of Logic and Algebraic Programming*, 58(1–2) :89–106, 2004. `http://krakatoa.lri.fr`.

[62] Laurent Mazaré. *Computational Soundness of Symbolic Models for Cryptographic Protocols*. PhD thesis, Institut National Polytechnique de Grenoble, France, October 2006.

[63] T.Furon M.Barni, F.Bartolini. A general framework for robust watermarking security. *Signal Processing*, 83(10) :2069–2084, October 2003.

[64] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer-Verlag, 2005.

[65] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proceedings of the Theory of Cryptography Conference*, pages 133–151, 2004.

[66] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the abadi-rogaway language of encrypted expressions. *Journal of Computer Security*, 12(1) :99–130, 2004.

[67] Carroll Morgan and Annabelle McIver. pGCL : formal reasoning for random algorithms. *South African Computer Journal*, 1999.

[68] Sungwoo Park, Frank Pfenning, and Sebastian Thrun. A probabilistic language based upon sampling functions. In *ACM Symp. on Principles of Programming Languages*, pages 171–182, 2005.

[69] Christine Paulin-Mohring. A library for reasoning on randomized algorithms in Coq. Description of a Coq contribution, Univ. Paris Sud, January 2006. `http://www.lri.fr/~paulin/ALEA/library.pdf`.

[70] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4) :511–540, 2001.

[71] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems. In *Workshop on Secure Architectures and Information Flow*, volume Electronic Notes in Theoretical Computer Science (ENTCS) 32, 2000.

[72] T. Rezk. *Verification of confidentiality policies for mobile code.* PhD thesis, Université de Nice Sophia-Antipolis, 2006.

[73] A. Sabelfeld and A. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Comunications*, 21 :5–19, January 2003.

[74] Andrei Sabelfeld and David Sands. Declassification : Dimensions and principles. *Journal of Computer Security*, 2007.

[75] Shmuel Sagiv, editor. *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3444 of *Lecture Notes in Computer Science*. Springer, 2005.

[76] Bruce Schneier. *Applied Cryptography.* John Wiley and Sons, 1996.

[77] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Vesta : A statistical model checker and analyzer for probabilistic systems. In *In 2nd International Conference on the Quantitative Evaluation of Systems, IEEE, (Tool paper)*, 2005.

[78] Victor Shoup. Sequences of games : a tool for taming complexity in security proofs, 2004. URL : `http ://eprint.iacr.org/2004/332`.

[79] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *SIGMOD*, 2003.

[80] J. Stern. Why provable security matters ? In E. Biham, editor, *Proceedings of EUROCRYPT'03*, volume 2656 of *LNCS*, pages 449–461. Springer-Verlag, 2003.

[81] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2002.

[82] S. Tarento. *Formalisation en Coq de modéles cryptographiques idéalisés et application au cryptosystéme ElGamal.* PhD thesis, Université de Nice Sophia-Antipolis, 2006.

[83] Sabrina Tarento. Machine-checked security proofs of cryptographic signature schemes. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, volume 3679 of *Lecture Notes in Computer Science*, pages 140–158. Springer, 2005.