1. EXECUTIVE SUMMARY ........................................................................................................... 2
2. CONTEXT AND RELEVANCE TO THE CALL ........................................................................ 4
  2.1. Context, economic and societal issues .............................................................................. 5
  2.2. Relevance of the proposal .................................................................................................. 6
3. SCIENTIFIC AND TECHNICAL DESCRIPTION ...................................................................... 8
  3.1. State of the Art ......................................................................................................................... 8
  3.2. S & T objectives, progress beyond the state of the art .......................................................... 11
4. SCIENTIFIC AND TECHNICAL OBJECTIVES / PROJECT DESCRIPTION ............................. 13
  4.1. Scientific Programme, Project structure .............................................................................. 13
  4.2. Project management ............................................................................................................... 14
  4.3. Description of the tasks.......................................................................................................... 15
    4.3.1 Task 1: Project coordination .......................................................................................... 15
    4.3.2 Task 2: Simple, automated proofs with strong, computational guarantees .................. 15
    4.3.3 Task 3: Proved implementations of protocols ................................................................. 18
    4.3.4 Task 4: Compose and decompose protocols and proofs: get modular proofs of security . . . 21
    4.3.5 Task 5: Logic for computational indistinguishability ......................................................... 23
  4.4. Tasks schedule, deliverables and milestones ......................................................................... 24
5. DISSEMINATION AND EXPLOITATION OF RESULTS. MANAGEMENT OF INTELLECTUAL PROPERTY .................................................................................................................. 26
6. CONSORTIUM DESCRIPTION .................................................................................................. 27
  6.1. Partners description & relevance, complementarity ................................................................. 27
1. EXECUTIVE SUMMARY

There is nowadays a huge quantity of information that flows through untrusted channels. And this will still increase dramatically in the future. Mobile sensor networks are deployed everywhere and control more and more social activities: car driving, health care, traffic control, ... Mobile and ad hoc networks, iPhone and more generally mobile terminals are increasingly used in everyday life. Chips are used to store sensitive data and are supposed to secure critical transactions: they are embedded in electronic passports, cash cards, ...

Securing the communications is therefore an important challenge. The research community in computer and information sciences has deployed a lot of efforts in trying to secure the communications. There is also a lot of efforts, in trying to increase our confidence by providing with "security proofs". Such proofs require a formal model for the protocols, for the security properties and for the attacker capabilities. Until 2001, different research communities worked independently, using different formal models. This includes for instance a logical (or symbolic) model and analysis, whose successes can be illustrated by the numerous man-in-the-middle attacks that have been found on security protocols. A second approach is computational, inheriting from complexity theory; it assumes for instance that the attacker is any randomized polynomial time Turing machine. This is the favorite model of cryptographers. A third approach is more pragmatic and tries to prove/find attacks on actual implementations of the protocols (this has been more successful in finding attacks than in proving protocols secure).

The main goal of the PROSe project is to provide with security proofs at several levels: symbolic, computational, and implementation.

In a nutshell, the aim of the project is to develop automatic protocol verification tools in the symbolic model, in the computational model, and in an implementation model. We will rely on the partner's experience in the areas of symbolic security proofs and their relationships with computational proofs.

Goals and challenges. The automation of proofs of protocols in the computational model, and its relationships with the symbolic model, have been investigated in the ANR project FORMACRYPT. Within this project, the tool CRYPTOVERIF has been developed and is now widely used in the world for computational proofs of some (small size) protocols. Several soundness results, that show the full abstraction of symbolic models, have been obtained, enabling the use of symbolic theorem provers for computational proofs.
These results and tools need however to scale up. The goal of the project is to broaden their application scope, for instance to protocol implementations.

There are several challenges that have been identified in each model:

- In the symbolic model, the verification tools such as AVISPA, PROVERIF, HERMES that have been developed by the partners of the project, do have some limitations. The main restriction concerns the class of protocols that can be considered. One (secondary) goal of the project is to overcome these restrictions.
- In the computational model, there are several challenges. First, the full-abstraction of symbolic models has actually a very limited scope, which restricts the applicability of the method. For instance, the current soundness results assume that a key is always generated using the key generation algorithm. While this is a reasonable assumption for a honest participant, it is a quite strong restriction concerning the attacker. Second, we are at a point where the computational soundness proofs become extremely complex, because of the possible interactions of cryptographic primitives. The main challenge is to design a modular proof strategy, allowing us to combine soundness results. This would enable handling more primitives and more protocols. Third, the CRYPTOVERIF tool is mainly a mechanization of the cryptographic game transformations. But is there a way to specify this at a logical level? Can we design a proof system that would prove computational indistinguishability properties?
- At the implementation level, there are currently very few works. A challenge is to produce programs that implement security protocols, while being formally proved. This would complete the chain of proofs from symbolic proofs to implementations.

Work Programme. The project PROSE is divided in four main tasks (in addition to the coordination task, task 1) that address the challenges:

Task 2 consists in extending the scope of the results of FORMACRYPT, typically to larger classes of protocols and properties. This includes extensions of the results in the symbolic model, but mainly extensions of CRYPTOVERIF and of soundness proofs, for instance allowing keys that are dishonestly generated by the attacker. The latter may require to us revise the symbolic model, hence to revise the symbolic automatic verification tools. That is why the two subtasks for symbolic proofs and for soundness results are inter-dependent.

Task 3 consists in deriving secure implementations from computational security proofs. In other words, we need to design, implement, and prove correct a compiler from specifications in a process algebra to an actual programming language (Ocaml). This compiler will assume that cryptographic primitives are secure, but we will also consider the proof of cryptographic primitives at the code level.

Task 4 consists in designing a modular framework for security proofs, and for soundness proofs. This requires in particular designing adequate abstractions that are suitable for a combination, without assuming too much on the security primitives.

Task 5 consists in designing a logic for indistinguishability, and then implementing it on top of a theorem prover. A grand challenge here would be to get some completeness result: while CRYPTOVERIF performs computational proofs, it is by no mean complete.
Impact. Altogether, the project outcome should be a series of results and prototype implementations that allow one to perform proofs of protocols at a larger scale, and include proofs of security for their implementations. This project is purely academic, though we will also associate M. Abadi and C. Fournet (Microsoft Research) and D. Lubicz (DGA) with whom we collaborated in the past. Nevertheless, we expect the outcome of the project to enable real-world applications; our results would give us an expertise and tools that can be used for (dis)proving critical security applications in all the areas that were described in the summary. In other words, our project does not include any industrial valorisation, but, if we reach our goals, we will be able to perform larger scale automatic security proofs and our tools will be applicable to a larger class of protocols; this will broaden the application area of our formal methods. Competing projects in the world are conducted at Saarbrücken (M. Backes) and at Stanford/CMU (J. Mitchell, A. Datta), with similar goals. But we believe that our consortium is in a good position to contribute in a significant way to the challenges.

2. Context and relevance to the call

The security of communications has become an ubiquitous challenge, because of the huge range of its applications. Security protocols and cryptographic primitives are typically designed to achieve security goals, despite an unreliable network. There is a wide range of possible attacks on communications, which we split in three categories:

- **Logical attacks**: they rely on protocol design mistakes; they are independent of the actual cryptographic primitives or implementations. Typical examples are the man in the middle attacks, whose numerous examples are reported in, e.g., [1].
- **Computational attacks**: they rely on weaknesses of cryptographic primitives, sometimes in combination with the logical part above. Such attacks typically occur when the cryptographic primitives are not adequate with the protocol (see [76] for instance).
- **Attacks on the protocol implementations**: they rely on programming mistakes, sometimes exploiting inadequacies between the computational assumptions and the actual implementations.

Each attack category corresponds to a protocol model. The main characteristic of a model is the class of attackers it allows and their computational power. For instance, the logical model excludes the possibility to break the cryptographic primitives. While the computational model does allow attacks based on weaknesses of the underlying cryptographic primitives, it abstracts from their implementation and the implementation of the protocol.

In view of the threats of security breaches, we need to increase as much as possible our confidence in the protocols. Testing is not possible, since an attacker will always choose the worst situation. That is why formal proofs and formal methods are crucial in this area. There has been a lot of efforts during the past two decades, in formally proving that a protocol is not subject to logical attacks. For instance the tool PROVERIF [29] allowed to derive several protocol security proofs. There has been also a lot of efforts devoted to formally proving the security of cryptographic primitives, by reducing any attack to a presumably hard
computational problem [17]. Finally, there has been also a lot of efforts in designing and proving the correctness of implementations of security primitives.

There is however one main issue, one major problem, which is our challenging task in this project:

**Goal:** Get formal security proofs at all levels: symbolic, computational, implementation

Indeed, the three approaches to security (logical, computational, implementation) have been mostly conducted concurrently, without much interaction. As a result, we could be faced with a protocol that has been proved secure in a formal system, while an attack is found later on. (This happened for instance with a protocol by Bull [70]). The reason is that the protocol proof is carried out in a model, which is different from the one in which an attack is mounted. It is therefore crucial to completely clarify the assumptions under which (and in which sense) a model is valid and to obtain security results in the most concrete model possible, which is the goal of our project.

### 2.1. **Context, Economic and Societal Issues**

With the growing development of Internet and electronic communications, the need for securing communications becomes more and more critical in many fields. Examples include:

- **Electronic commerce:** according to FEVAD figures (the French federation of distance selling companies), e-commerce turnover in France has increased by 20% in 2008, to reach 20 billion euros and represent about 80% of all distance selling [46], so its economic importance is unquestionable. Attacks against e-commerce communications can lead to the theft of credit card numbers, leading to fraud, loss of money, and also loss of confidence from customers, which could delay or prevent the adoption of electronic commerce. In this example, the confidentiality of the communication must be guaranteed, but also authentication, so that the client is assured of the identity of the merchant and conversely.

- **Privacy:** with the ubiquity of electronic services, more and more information about the people’s activity and private life is released, so there is a growing demand for services that preserve the privacy. Privacy is also a legal requirement, for example by *loi informatique et libertés* (informatics and liberty law) in France or by the European directive on privacy and electronic communications. Privacy requires properties such as anonymity, which is a more complex security property than secrecy and authentication. Considering such properties is one of the goals of our project, since we believe that privacy is becoming a key issue.

- **E-voting** is an example is which what is at stake is of obvious importance. Moreover, using electronic means may make it easier to mount a very large fraud, for example by casting many false bulletins using a computer program. In countries such as the US (or even France to a much smaller scale) in which e-voting is starting to be used for political elections, the lack of security of the voting machines has been
pointed out [54]. The security properties required for e-voting are particularly complex (anonymity of voters, verifiability of the results, ...). While they start to be understood in the symbolic model [45], there is still a lot of work to do to formalize and prove them in the computational model using formal methods. Our project aims to go in this direction, even if we consider it as a long-term goal.

Moreover, nowadays, most attacks against practical security protocols rely on implementation-specific details, so they fall outside the basic logical and computational models. For instance, such attacks were found against Google sign-on [7] and the implementation of Kerberos in Windows [2]. This is why our project aims to provide proofs also at the implementation level.

2.2. Relevance of the Proposal

The formal verification of security protocols, in particular the verification of implementations and the proofs in the computational model, is a very active research topic, as shown by recent publications in conferences such as CSF or workshops such as FCC.

At the national and bi-national levels, there are three related projects in which some of the members are involved:

- the ANR AVOTÉ project (SeSur 2007) on the formal analysis of electronic voting protocols. One of the workpackages of this project aims at extending computational soundness results to properties and primitives used in electronic voting protocols. The project focuses on electronic voting and does neither consider general proof techniques for soundness results, nor direct computational proofs, nor guarantees at the implementation level.
- the ANR SCALP project (SeSur 2007) on the Security of Cryptographic ALgorithms with Probabilities. The aim of this project is to develop computer-aided proof methods for the security of cryptographic primitives and protocols. The focus is on cryptographic primitives and on computational attacks. In particular, the project does not consider implementation-based attacks.
- the CNRS/JST French-Japanese collaboration project on Computational and Symbolic proofs of security (CoSyProofs). The scientific goals of CoSyProofs are similar, though less precise. The CNRS/JST support is dedicated to organizing visits and joined workshops with the Japanese researchers.

At the international level, there are four important related projects, with which we will compare our goals and assets along the project description:

- PCL (J. Mitchell, Stanford & A. Datta, CMU). Roughly, the goals are similar, but the approach is very different, as PCL is trying to design a Hoare-style logic to (manually) prove protocols.
END2END Security (M. Backes et al, Saarbrücken). This group designed its own model. Attempts to formalize the proofs using Isabelle are reported in [73]. Recent works such as [8] aim however at unifying the models and constructing modular proofs, which is also our goal.

CertiCrypt (G. Barthe et al, Madrid). This project aims at formalizing the games-based technique for proving security protocols in the computational model using the general-purpose theorem prover Coq. The games-based technique is essentially a notation that allows to structure security proofs. This approach is very flexible, but requires much more user interaction than the techniques we will develop in this project. The CertiCrypt tool is currently developed within the SCALP project, see above.

C. Fournet’s project (Microsoft Research and MSR-INRIA lab). This project develops several approaches for the verification of protocol implementations, by extraction of protocol specifications or by typing, in the symbolic and in the computational models.

Individual researchers are also contributing to the area, with similar goals, for instance R. Segala (Verona) and B. Warinschi (Bristol).

The current project also builds on a former project, FORMACRYPT, which was supported by the ANR (January 2006-July 2009). The FORMACRYPT project aimed at relating the symbolic and computational models and at providing automated proofs of protocols in the computational model. It reached these goals, in particular by providing several computational soundness theorems, which show that a security result in the symbolic model implies a similar result in the computational model, and two tools, a module in the symbolic verifier AVISPA that provides proofs in the computational model by relying on a proof in the symbolic model and a computational soundness theorem, and the verifier CRYPTOVERIF, which proves protocols directly in the computational model. The current project goes considerably further in several directions: in particular, we will make computational soundness proofs modular, which is necessary to handle many cryptographic primitives simultaneously; we will consider proofs at the level of implementations and not only specifications.

This project fits in topic 5 of the VERSO call “security, confidence in communications”. Formally proved security protocols, as we aim at providing in our project, are a key component of communication security. As mentioned in the call, the compromise of sensitive data can have a huge economical, political or even military impact, and we believe that the only way to obtain actual security guarantees in order to avoid such compromise is by proofs, because tests cannot consider all attack scenarios. Formal methods are a particularly attractive approach to perform these proofs because they offer the possibility of automating or at least automatically verifying these proofs, therefore making them much more reliable than manual proofs, which are particularly error-prone.
3. **Scientific and Technical Description**

3.1. **State of the Art**

To be practically relevant to protocol designers and system developers, existing automated verification methods for security protocols have to be enhanced in several ways.

Built-in versus user-defined security properties: There are several automated tools that check whether a given model (a process) satisfies a security property: AVISPA, FDR, Scyther, NRL Protocol Analyzer, ... Most of these verification tools are able to check built-in security properties, essentially confidentiality and some authentication properties. Indeed, they are based on reachability analysis, and hence, are limited to trace properties. Although, these properties are very useful, most modern security protocols aim at establishing properties that go beyond basic confidentiality and authentication. This is the case for group key exchange protocols, e-voting protocols, e-auction protocols, contract signing protocols, etc. A typical example of a property that one might want to check is *anonymity*, which requires two samples of the process: typically one in which A is related to vA and B is related to vB and another in which A is related to vB and B is related to vA. These two processes should be indistinguishable.

Indistinguishability, aka observational equivalence, of processes allows a user to define the security property he is interested in by defining an ideal process. A protocol is then deemed to be correct, if it is indistinguishable from the ideal process. Besides its expressive power (indeed, indistinguishability cannot be expressed as a trace property), indistinguishability is also more suitable for modular verification.

PROVERIF [29] is an exception among the security verification tools as it goes beyond trace properties by considering a process equivalence relation that can be checked on a single *bi-process* [30]. This relation is, however, stronger than process indistinguishability, and hence not suitable in some cases, typically for e-voting protocols where additional manual proofs are necessary [9, 45].

Hard-wired model assumptions: Another limitation from which existing automated verification tools suffer stems from the fact that they are limited to a fixed number of principals in each session, and do not apply to parameterized group protocols, in which the number of participants is a parameter. There are several works that solve special cases of group protocols [38, 56, 57] but for a bounded number of sessions [38, 57] or for passive adversaries (eavesdroppers) [56]. More generally, the expressiveness of the existing tools is limited by hard-wired limitations of the underlying attacker and protocol models. For instance, they do not consider adaptive corruption.

Maybe more importantly, the proofs that are completed using the above tools may be disputed since the attacker is only given a fixed set of primitives considered as black boxes and the messages are terms on these primitives. In a more realistic model, the messages are bitstrings, the primitives are functions on these bitstrings, and the attacker is allowed to
perform any probabilistic polynomial-time computation: this yields what we call the computational model.

In order to obtain proofs in this model, there are basically two approaches that have been followed since 2001: in the first one, we try to define a finite (fixed) set of primitives, together with computational assumptions on them, so that the symbolic model is sound with respect to the computational one. Soundness means that we faithfully abstracted all relevant operations that an attacker may ever perform; any other operation will be useless for mounting an attack (except with a negligible probability, for instance flipping a coin in order to guess a secret key). This approach, called computational soundness, started with the seminal work of Abadi and Rogaway [4, 5] and was extended in many respects since then (see for instance [41, 52, 53, 40, 39] or the series of papers by Backes et al [13, 14, 11]). This was also part of the ANR project FORMACRYPT (January 2006-July 2009) in which many of the partners of the current project were involved. Most works consider a single or a few cryptographic primitives: [41, 52] handle public-key encryption and signatures, [53, 40] add hash functions, [39] deals with shared-key encryption. These works also often use different protocol representations and formalisms, which makes unification difficult. The library of Backes et al considers many primitives simultaneously, but relies on a modified symbolic model (for instance, the length of messages is explicit in their model). Recently, a unifying framework for modular computational soundness proofs [8] appeared, but most proofs are not done yet in this framework ([8] considers only public-key encryption and signature).

All the abovementioned results make strong assumptions on the attacker’s capabilities. Let us mention for instance that the attacker is always supposed to use the key generation algorithm, when he is generating a new key. This is acceptable in the case of public-key cryptography, since one may assume that public keys come with a certificate, but is an unrealistic assumption in the case of symmetric encryption. There are other technical assumptions, for instance the ability to specify a length function in the symbolic model, which is not supported by current symbolic protocol verifiers, or the existence of a parsing function from bitstrings to terms. The latter rules out some cryptographic primitives such as exclusive or, for instance [12]. The soundness proofs also required new assumptions, some of which might be not needed or may be dropped by adding new primitives. (A typical example is the key cycle problem).

The second approach for obtaining proofs in the computational model is more direct. It simply ignores the most abstract models and conducts proofs directly in a computational model. Since formal proofs at a lower (more detailed) level are quite involved, this requires the assistance of a theorem prover. That was exactly the purpose of CRYPTOVERIF [26, 27, 28], a specialized theorem prover that allows one to perform security proofs at the computational level. It was developed within the ANR project FORMACRYPT and, again, it was quite successful in proving several protocols in this model [32, 31, 21]. More recently, Laud and Tšahhirov [75, 62] built a similar tool, however limited to public-key and shared-key encryption. Another approach, CERTICRYPT [15, 16, 18], relies on the general-purpose theorem prover Coq to perform proofs in the computational model. This approach offers
more flexibility at the cost of heavy user interaction. Computationally sound type systems [61, 10] can also be used to prove protocols in the computational model, with strong assumptions on the cryptographic primitives. Logics may also be used to prove protocols directly in the computational model. For instance, PCL (Protocol Composition Logic) [42], which was initially designed for the symbolic model, was extended to the computational model [43].

The two approaches: direct security proofs and symbolic proofs + computational soundness are currently competing. As analyzed on a simple example [3], there is still a lot of work to do in the soundness approach, but in many respects the approaches are complementary. The current project includes representatives of both schools and we expect several improvements in this stimulating environment.

So far, we did not say anything about proofs of implementations. Again, this raises compatibility issues between the security models. Again, we could try either to perform direct proofs of the code or try to derive secure code from the security proofs at the computational level. Proving the code is a huge task. Abstraction techniques seem to require very heavy work and are still at their beginning. Goubault-Larrecq and Parennes [50] present an abstraction technique from C programs to the symbolic models. Bhargavan et al [25, 22, 24] extract symbolic models from ML programs and verify them using PROVERIF. They have recently adapted this approach to extract computational models and verify them using CRYPTOVERIF [20]. They have applied both techniques to the TLS protocol [21], but verify a much smaller part of the protocol in computational model than in the symbolic model. Much work is still needed in order to get a complete proof of this protocol in the computational model. (Actually, corrections to the protocol may also be required [37].) Recently, Bhargavan et al also introduced a typing approach to verify implementations of security protocols in the symbolic model [19, 23]. This approach is more scalable than their previous global analysis approach, at the cost of the manual addition of type annotations. The adaptation of this approach to the computational model is at its very beginning [47]. We plan to investigate within the project the other approach that consists in deriving secure implementations from security proofs, namely proofs using CRYPTOVERIF. This approach was already considered in the symbolic model [72, 64, 69, 74, 51, 67, 68, 66], but not yet in the computational model.

Finally, important issues that are part of the project concern the scope of the methods. Concerning scalability, we need to be able to perform modular security proofs. This was the motto of universal composability [34], that has been very popular in the cryptographic community these last years, and of Protocol Composition Logic [42, 44]. Proofs of universal composability are however quite hard and error-prone and, also, require stronger assumptions on the primitives. They might also be obtained by combining modularity results at a more abstract level (for instance using the result of [6]) and computational soundness results. Broadening the scope also involves considering more general processes and more security properties, which is part of our project.
3.2. S & T OBJECTIVES, PROGRESS BEYOND THE STATE OF THE ART

The ultimate goal of our project is to obtain proofs of security protocols at all levels: symbolic, computational, and at the level of implementations. In order to reach this goal, we consider several directions.

First, we will extend the existing techniques for the proof of protocols at the computational level, in the line of the FORMACRYPT project, but obviously going further. We will consider both direct computational proofs and proofs via symbolic techniques combined with computational soundness theorems. This will yield new symbolic challenges, in order to obtain the symbolic results needed to apply our computational soundness theorems. In particular, we wish to investigate the group protocols and the automation of indistinguishability proofs, beyond what is currently performed in PROVERIF. At the computational level, we plan to extend both direct techniques and computational soundness results to new primitives, such as exclusive or, Diffie-Hellman key agreements, and blind signatures, which are useful for e-voting protocols. We also wish to weaken the assumptions of the soundness results, so as to get more realistic and relevant assumptions for the validity of the symbolic model with respect to the computational one. This will probably require us to add some intruder capabilities to the symbolic model, which, in turn, triggers new challenges for the symbolic verification tools. A typical target is to allow dishonest key generation.

We also plan to pursue three novel research directions. First, we would like to obtain proofs at the level of implementations. Indeed, even if a specification is proved secure, new attacks may appear at the implementation level, because they exploit programming mistakes. The main obstacle to reach this goal is to make proofs at the level of a full programming language, and not only at the level of a fairly simple, specialized specification language. To reach this goal, we plan to implement a proved compiler that translates a specification (proved correct at the computational level), to an implementation of the protocol. However, this approach proves the protocol assuming that the cryptographic primitives are secure, so we will complement it with an approach for proving cryptographic primitives at the code level, by relying on assertions in the style of Hoare logic.

Second, we would like to obtain modular security proofs. Indeed, in general, the protocols are not considered in isolation. For instance, a protocol will rely on public-key cryptography, assuming a public-key infrastructure. More specifically, when we try to prove formally some complex protocols such as E-voting protocols, we are soon faced with complex combinations of security primitives, each of which assuming that some data exchange has been securely completed before. A soundness proof, or a proof in the computational model, becomes then extremely complex, and cannot be managed without intermediate subgoals. The very same reasons justified the introduction of universal composability (UC) [34, 36, 58, 60]. Though realizable, the UC properties of basic security primitives are quite restrictive and, for protocols, UC proofs require the design of ideal functionalities, which is not always possible [63]. Even when this is possible, UC proofs for protocols are extremely hard and have been completely carried out in very restricted cases only [35]. On the other hand, there are very
simple similar results in the symbolic setting [6]. The criteria can be checked quite easily. Transferring them to the computational model requires however soundness results, as explained above. This poses the problem of composing soundness results. This problem also raises when conducting soundness proofs for complex combinations of primitives, for instance in the framework of e-voting protocols. Proofs become too complex and require to be split. This yields our next challenge: design a framework for modular proofs of soundness.

Third, we plan to design a logic for reasoning on computational indistinguishability in the concrete security framework, in which one can obtain explicit formulae that bound the probability of an attack. We also plan to implement this logic on top of a theorem prover such as Coq. We will prove its soundness and investigate its completeness. This will provide a novel, flexible approach for mechanized computational proofs of protocols.

Our project will produce both theoretical results, such as soundness theorems (which are also applicable in combination with symbolic protocol analysis tools such as AVISPA or PROVERIF), and several tools and prototypes. In particular, we will produce a new, extended version of CRYPTOVERIF, our prover of protocols in the computational model. We will also produce a compiler from protocol specifications to implementations and an implementation of the new computational indistinguishability logic on top of a theorem prover.

The main success criteria for our project will be publications in the main conferences of the field (such as CSF, IEEE Symposium on Security and Privacy, ESORICS, CRYPTO) and the results of our techniques and tools on protocol benchmarks.

In the long term, our goal is to obtain formally proved implementations of practical security protocols in the computational model, which is very ambitious. For instance, a grand challenge (worldwide) would be to formally prove some crux protocol, such as a widely used protocol such as Kerberos or TLS, or an online E-voting protocol, in the computational model and to derive a secure implementation. The social impact of such a proof would be huge. Such a challenge is beyond the scope of our project. But our project will contribute to that ultimate goal, by providing with the necessary building blocks: proofs in computational models, deriving secure implementations and modularity of proofs. Furthermore, these building blocks can be used not only for proving a particular protocol, such as an E-voting protocol, but also for any secure communication purpose. As a result, we will be able to prove and get secure implementations of intermediate size protocols.

Such a task, that involves fundamental research and whose goal is long term, can only be conducted within the Academic world. We also believe that it has to be eventually completed, because there is a need for security protocols (for instance for e-commerce or online E-voting) and, whether proven secure or not, they will eventually be deployed. In view of the stakes, it is better if they are proved secure. Moreover, even if our research is academic, the results that we produce are readily applicable outside the academic world by using our tools.
4. **SCIENTIFIC AND TECHNICAL OBJECTIVES / PROJECT DESCRIPTION**

4.1. **SCIENTIFIC PROGRAMME, PROJECT STRUCTURE**

The main goal is to get automated security proofs of protocols at different levels of abstraction. To reach this goal, our work will be split into the following tasks:

1. Project coordination
2. Simple, automated proofs with strong, computational guarantees:
   2.1. Extension of the symbolic methods to more primitives and protocols
   2.2. Extension of soundness results, broadening the application scope of symbolic proofs
   2.3. Extend direct, automatic, computational proofs
   2.4. Case study
   2.5. Comparison of the approaches
3. Proved implementations of protocols.
   3.1. Generation of implementations from CRYPTOVERIF specifications
   3.2. Hoare style proofs for cryptographic primitives
   4.1. Deducibility/indistinguishability games
   4.2. Relevance of the games with respect to assumptions
   4.3. Relevance of the games with respect to soundness proofs
   4.4. Modularity results
   4.5. A computational proof of FOO
5. Logic for computational indistinguishability.
   5.1. A model for cryptographic systems
   5.2. A formal logic for reasoning on this model
   5.3. Proof search algorithms and heuristics
   5.4. Implementation

These tasks are described in more detail below. Task 2 concerns the symbolic and computational proofs of security, following the techniques that have been already investigated by the partners. The three other tasks investigate new goals and new techniques in three different directions: deriving programs, modularising the proofs, and applying classical theorem proving to computational/information theoretic proofs.

Task 2 consists in extending previous work in three directions: symbolic proofs, computational soundness results (which yield computational security proofs by combining them with symbolic proofs), and direct computational proofs. These approaches will be applied to case studies and compared. Task 3 aims at proving implementations: we will derive secure implementations of protocols assuming primitives are secure, and also prove the security of primitives at the code level. Task 4 aims at getting modular proofs, which are necessary for scalability. Task 5 aims at providing a logic for reasoning about computational indistinguishability and investigating proof search algorithms that can be implemented on top of a theorem prover-based formalization of the logic.
The next figure illustrates the links between the tasks.

![Diagram of task links]

Task 2, which extends computational proofs, will be useful in order to perform task 3, that generates implementations from a proved specification. Task 4 and task 2 will be mutually helpful: task 4 will help make computational proof scale up, while task 2 will provide basic building blocks that can be combined using the modularity results of task 4. Task 5 provides a novel, flexible way of obtaining computational proofs.

All these tasks include activities for dissemination and valorization, in particular by publication of research papers in conferences and distribution of software on Internet.

### 4.2. Project management

The project coordination corresponds to task 1 in the financial document.

We will organize two plenary meetings per year. We will invite researchers from outside the project to these meetings, to improve the visibility of the project at the international level and the communication with other teams. (We did this for the FORMACRYPT project and it proved very successful.) In particular, Martin Abadi, Cédric Fournet (both from Microsoft research), and David Lubicz (CElar, DGA) will be privileged partners. Martin Abadi is a pioneer of the field, Cédric Fournet is also a well-known expert of the topic. Moreover, Cédric Fournet and David Lubicz are users of our tools. Some of these meetings will be joint with the workshop FCC (workshop on Formal and Computational Cryptography), which fits exactly the topic of this project. We plan the following schedule for plenary meetings:

- December 2010: kickoff meeting, Paris
- July 2011: joint meeting with CSF/FCC organized by LSV
- Fall 2011: project meeting in Nancy
- Spring 2012: project meeting Grenoble
- Fall 2012: project meeting in Paris
- Spring 2013: project meeting in Cachan
- Fall 2013: project meeting in Nancy
- Spring 2014: project meeting in Grenoble
- Fall 2014: final meeting in Paris.
The meetings of spring 2012, 2013, 2014 may in fact take place in July joint with CSF/FCC if the project members go to this conference and workshop to present their work, which is fairly likely given our research topic. Smaller meetings between several members of the project will be organized more often if needed.

We will also produce a progress report every year, and maintain a web page of the project.

The project consists of 4 tasks (in addition to the coordination task); each of the four partners will be responsible for one task (which will obviously be performed in collaboration with the other partners).

4.3. DESCRIPTION OF THE TASKS

4.3.1 TASK 1: PROJECT COORDINATION

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Project coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>T0</td>
</tr>
<tr>
<td>End</td>
<td>T0+48</td>
</tr>
<tr>
<td>Duration: 48 months</td>
<td>Task leader: Bruno Blanchet (INRIA Paris-Rocquencourt)</td>
</tr>
</tbody>
</table>

Objectives: Task 1 guarantees that the project evolves as planned

Description:
- Define and schedule tasks within the project
- Organization of project meetings
- Management of the project web page
- Supervision of the progress of the work
- Production of progress reports

<table>
<thead>
<tr>
<th>Deliv.</th>
<th>Ver.</th>
<th>Title</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1.1</td>
<td>Final</td>
<td>Progress report after the first year</td>
<td>T0+12</td>
<td>Document</td>
</tr>
<tr>
<td>D1.2</td>
<td>Final</td>
<td>Progress report after the second year</td>
<td>T0+24</td>
<td>Document</td>
</tr>
<tr>
<td>D1.3</td>
<td>Final</td>
<td>Progress report after the third year</td>
<td>T0+36</td>
<td>Document</td>
</tr>
<tr>
<td>D1.4</td>
<td>Final</td>
<td>Final report on project results</td>
<td>T0+48</td>
<td>Document</td>
</tr>
</tbody>
</table>

Validation criteria: Availability of the deliverables

Resources:
- INRIA Paris-Rocquencourt 4 mm.
- LSV 1 mm.
- LORIA 1 mm.
- VERIMAG 1 mm.

This task ensures the smooth evolution and the consistency of the project. In particular, it organizes one plenary meeting every six months, as detailed in Section 4.2.

4.3.2 TASK 2: SIMPLE, AUTOMATED PROOFS WITH STRONG, COMPUTATIONAL GUARANTEES

<table>
<thead>
<tr>
<th>Task 2</th>
<th>Simple, automated proofs with strong, computational guarantees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>T0</td>
</tr>
<tr>
<td>Duration: 48 months</td>
<td>Task leader: Hubert Comon-</td>
</tr>
</tbody>
</table>
**Objective**
Extend existing techniques for computational proofs of protocols

**Description**
See detailed description below

**Deliverables and Versions**

<table>
<thead>
<tr>
<th>Deliv.</th>
<th>Ver.</th>
<th>Title</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2.1</td>
<td>Final</td>
<td>Survey of soundness results</td>
<td>T0+12</td>
<td>Document</td>
</tr>
<tr>
<td>D2.2</td>
<td>1</td>
<td>A version of CRYPTOVERIF that handles DDH</td>
<td>T0+12</td>
<td>Software</td>
</tr>
<tr>
<td>D2.3</td>
<td>Final</td>
<td>A prototype implementation for task 2.1.2</td>
<td>T0+24</td>
<td>Software</td>
</tr>
<tr>
<td>D2.4</td>
<td>2</td>
<td>A version of CRYPTOVERIF that handles XOR</td>
<td>T0+36</td>
<td>Software</td>
</tr>
<tr>
<td>D2.5</td>
<td>Final</td>
<td>A page reporting the results of our tools on benchmarks</td>
<td>T0+48</td>
<td>Document</td>
</tr>
</tbody>
</table>

**Validation criteria**
Publications, availability of software, results of our tools on benchmarks

**Resources**

<table>
<thead>
<tr>
<th>LSV</th>
<th>INRIA Paris-Rocquencourt</th>
<th>LORIA</th>
<th>VERIMAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.4 mm.</td>
<td>54 mm.</td>
<td>33 mm.</td>
<td>8 mm.</td>
</tr>
</tbody>
</table>

**Task 2.1: Extensions of symbolic methods**
New protocols, new security properties, but also the computational soundness proofs, bring new challenges for the symbolic proof techniques. We have three specific goals:

1. **Group protocols**: we wish to extend the tool PROVERIF, so as to be able to verify protocols that include an unbounded number of participants.
2. **Observational equivalence properties**: we plan to extend the algorithm for solving deducibility constraints, to proofs of equivalence properties. Such an algorithm should be implemented in a prototype tool and the outcome should be compared with the current cases that are covered by PROVERIF.
3. **Extensions of attacker capabilities**: we may have to modify the attacker model, according to the soundness proofs of task 2.2. There are two typical examples. First, we certainly need to reason on the (formal) length of messages, which is not possible currently in symbolic tools. Next, the ability to generate fake keys (as described below) will force to modify the symbolic model, as an attacker may forge a key that has additional (symbolic) properties.

**Task 2.2: Extensions of soundness proofs.**
Currently the soundness proofs are disappointing because of several limitations that we plan to wave:

1. **Dishonest keys.** The first need is to relax the assumptions for symmetric encryption (an obviously widely used primitive). Indeed, the current state-of-the-art (including [14]) provides soundness results only when the attacker generates his keys according to the encryption scheme (or provided that unique session identifiers are added in each message [59]). There is however no way to prevent a malicious user from computing his keys at his advantage, unless a trusted party is used, which is at least inefficient and sometimes even impossible given the infrastructure.
Our first goal is to obtain a soundness result, still allowing the adversary to compute his keys at his will. We think that it is possible to take advantage of the fact that, usually, keys are used for encryption only after a first phase where there is already some authentication between the agents. We may also strengthen the security hypotheses on encryption schemes, requiring e.g. that a ciphertext cannot be successfully decrypted with different keys, even for dishonestly generated ones.

2. Exclusive or (at least in the context of security APIs). There is currently no soundness result for the exclusive or, a widely used primitive. Actually there are impossibility results for exclusive or in the simulatability framework [12]. This discouraged research in this direction. However, impossibility results are strongly related to the particular method that is used. It seems indeed that the impossibility is due to a commitment problem, itself due to a parsing requirement: in the simulatability approach, there should exist a simulator, who is able to convert terms to bitstrings and bitstrings to terms. Other approaches to soundness proofs do not rely on such a simulator. Therefore, there is a hope to bypass the impossibility result.

3. Other primitives. Protocols use several other primitives; for instance e-voting protocols involve blind signatures and re-encryption, for which no soundness result is known. The reason for this lack of results is that such primitives are never used in isolation: they are combined with encryption or signatures for instance. Building a soundness proofs for several cryptographic primitives at the same time is however very painful. Relying on Task 4, we could however compose the soundness proofs and hence consider the above primitives in isolation.

4. Weakening security assumptions. Existing soundness results make use of strong security hypotheses. For example, encryption schemes are typically assumed to be IND-CCA2, which is not the case of all schemes used in practice (e.g. in security APIs), for efficiency reasons. We think that it is possible to weaken the security hypotheses on primitives, exploiting the fact that protocols do not exchange arbitrary type of messages. For example, there is often some freshness check (on a nonce, which is a randomly generated number). This may suffice for re-establishing cryptographic guarantees, under weaker assumptions on the primitives.

Task 2.3: Extension of direct, automatic, computational proofs
We are extending the automatic prover CRYPTOVERIF which produces proofs by sequences of games, directly in the computational model. We plan the following extensions:

1. Extension to Diffie-Hellman key agreements. CRYPTOVERIF currently supports the computational Diffie-Hellman assumption (CDH); we will extend it to the decisional Diffie-Hellman assumption (DDH), which requires an extension of the language that we use to specify the security assumptions on primitives. We will apply this work to practical protocols, such as the variants of the password-based protocol EKE [33], the mode of Kerberos that relies on the Diffie-Hellman key agreement, or a simplified version of SSH.

2. Extension to exclusive or (XOR). Currently, CRYPTOVERIF does not fully support XOR because associative function symbols cannot be encoded. We will extend it to support associativity. We plan to apply this extension to the OAEP scheme [71, 49] or its variants. (The proof of OAEP itself is probably too difficult to perform within CRYPTOVERIF in the
short term, as it relies on complex reasoning, but its variants OAEP+ [71] and OAEP 3-round [65] should be easier to tackle.)

The main risk in this task is that new forms of reasoning, beyond the addition of the primitives, may be needed to prove these protocols. We will add these forms of reasoning if possible, or otherwise fall back to simpler examples of protocols.

**Task 2.4: A case study**

We will apply symbolic methods, the soundness results, as well as direct automatic computational proofs to a case study of a simple e-voting protocol [48]. This proof requires in particular the treatment of blind signatures, which we will add to the various approaches.

**Task 2.5: Comparison of the approaches**

We will compare the direct approach and the approach through soundness results on a sequence of benchmarks, in the spirit of [3].

**Who does what**

<table>
<thead>
<tr>
<th>Task</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>LSV, INRIA Paris-Rocquencourt</td>
</tr>
<tr>
<td>2.2</td>
<td>LSV, LORIA, INRIA Paris-Rocquencourt, VERIMAG</td>
</tr>
<tr>
<td>2.3</td>
<td>INRIA Paris-Rocquencourt</td>
</tr>
<tr>
<td>2.4</td>
<td>LSV, LORIA, INRIA Paris-Rocquencourt</td>
</tr>
<tr>
<td>2.5</td>
<td>LSV, LORIA, INRIA Paris-Rocquencourt</td>
</tr>
</tbody>
</table>

---

### 4.3.3 Task 3: Proved implementations of protocols

**Task 3**

Proved implementations of protocols

<table>
<thead>
<tr>
<th>Start</th>
<th>End</th>
<th>Duration: 48 months</th>
<th>Task leader: Bruno Blanchet (INRIA Paris-Rocquencourt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>T0+48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Objectives**

Obtain protocol implementations proved in the computational model

**Description**

See detailed description below

<table>
<thead>
<tr>
<th>Deliv.</th>
<th>Ver.</th>
<th>Title</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3.1</td>
<td>Initial</td>
<td>Prototype of compiler from specifications to implementations, working on small protocols</td>
<td>T0+12</td>
<td>Software</td>
</tr>
<tr>
<td>D3.2</td>
<td>Final</td>
<td>Paper on the generation of implementations from specifications</td>
<td>T0+24</td>
<td>Document</td>
</tr>
<tr>
<td>D3.3</td>
<td>Final</td>
<td>Paper on the verification of cryptographic primitives</td>
<td>T0+24</td>
<td>Document</td>
</tr>
<tr>
<td>D3.4</td>
<td>Final</td>
<td>Enhanced compiler from specifications to implementations</td>
<td>T0+48</td>
<td>Software</td>
</tr>
<tr>
<td>D3.5</td>
<td>Final</td>
<td>A page reporting the results of our tools on benchmarks</td>
<td>T0+48</td>
<td>Document</td>
</tr>
</tbody>
</table>
Task 3.1: Generation of implementations from CRYPTOVERIF specifications

This task will consist in the implementation and proof of a specialized compiler that translates a CRYPTOVERIF specification of the protocol into an OCaml implementation. Using CRYPTOVERIF to prove the protocol in the computational model, this compiler will generate an implementation also proved in the computational model. Only the protocol itself will be coded in CRYPTOVERIF. The cryptographic primitives will be implemented directly in OCaml. (There already exists a cryptographic library for OCaml.) The CRYPTOVERIF specification makes it clear which security assumptions are made on each primitive. The code for interacting with the network will also be written directly in Ocaml; no security assumption needs to be made on this code: it can be considered as part of the adversary. An important advantage of OCaml as a target language is that it is memory safe, so we do not need to verify the absence of buffer overflows in the manually written code. A preliminary prototype is being implemented, but there is still much work to do before getting practical results.

1. Proving the compiler. Indeed, our goal is to obtain a proved implementation. CRYPTOVERIF can be used to prove security properties of the specification, but we need to prove that our compiler faithfully translates that specification. This proof essentially consists in establishing a correspondence between the semantics of the CRYPTOVERIF specification and the semantics of the generated OCaml programs. These semantics are fairly complex: protocols are probabilistic; furthermore, since we consider several OCaml programs running in parallel on several machines (for example, a client and a server), the semantics of the implementation includes non-determinism that we have to resolve, for instance by introducing an explicit scheduler.

2. Extending the input language to make it more expressive. In particular, the input language contains the protocol itself and a set of assumptions on how the protocol is used: the model. Currently, the protocol and the model are mixed in the input language, and separating them would help clarify the specification. Adding private channels could be a way to do that. It could also help separate implementation details from the protocol itself. We will also add a specific construct to model tables of keys often used in protocols. Currently, we can essentially specify loops only without internal state; we will add constructs to code more general loops. All these points will require both extensions of the compiler and extensions of CRYPTOVERIF. We also plan to design a common input language for PROVERIF (protocol verifier in the symbolic model), CRYPTOVERIF, and our compiler. Therefore, we will be able, from a single protocol specification, to verify it in the symbolic model and in the computational model, and to generate a proved implementation from it. We will obtain a complete platform for the design and verification of protocol specifications and for the generation of protocol implementations.
3. Applying the tool to case studies. We plan to apply it not only to protocols of the literature, but also to more ambitious case studies on practical protocols, such as FOO, Kerberos or TLS. These protocols are complex, so we may consider only certain modes of these protocols, and we may also need to correct the protocols so that they are actually provable.

**Task 3.2:** Hoare style proofs for cryptographic primitives

Task 3.1 focuses on protocol implementation on top of a library of cryptographic primitives. One major goal of the language and model envisioned in that task is to make the security assumptions on the used primitives explicit and precise. To complete the picture one needs to check that used primitives indeed meet the security assumptions. This is the aim of Task 3.2. More concretely, building on previous work where a toy programming language is used, we aim at developing automated verification procedures for cryptographic primitives at code level. Our approach will be based on Hoare logics built upon a carefully chosen assertion language that allows us to state properties useful for security proofs such as freshness, indistinguishability, and secrecy of values held by variables. This raises several challenges: so far, developed assertion languages are not suitable for verifying several widely used cryptographic primitives such as digital signatures and message authentication codes; more general verification procedures for symmetric encryption modes and hybrid encryption are also mandatory; one needs to integrate cryptographic verification with general purpose program verification techniques, which are making substantial progress; the underlying semantic model has to be matched with the model of Task 3.1.

1. Developing a Hoare style verification method. We need to develop an assertion language that captures the program properties of implementations of cryptographic primitives, axioms and rules for reasoning about the programs and sound (but incomplete) analysis procedure based on the Hoare logic. We plan to interface this with the proving compiler of Task 3.1 in such a way that the security of the implemented protocols can be stated precisely. We believe that other projects may benefit from such an analysis procedure. Simplistic prototypes for asymmetric encryption and symmetric encryption modes have been implemented and tested.

2. Case study. Analyse and prove correct an implementation of a set of primitives used by one of the case studies of Task 3.1 such that we get a completely verified implementation for the considered protocol.

**Who does what**

<table>
<thead>
<tr>
<th>Task</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>INRIA Paris-Rocquencourt, VERIMAG</td>
</tr>
<tr>
<td>3.2</td>
<td>VERIMAG, INRIA Paris-Rocquencourt</td>
</tr>
</tbody>
</table>
### 4.3.4 Task 4: Compose and Decompose Protocols and Proofs: Get Modular Proofs of Security

<table>
<thead>
<tr>
<th>Task 4</th>
<th>Compose and decompose protocols and proofs: get modular proofs of security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>T0</td>
</tr>
<tr>
<td>End</td>
<td>T0+48</td>
</tr>
<tr>
<td>Duration</td>
<td>48 months</td>
</tr>
<tr>
<td>Task leader</td>
<td>Véronique Cortier (LORIA)</td>
</tr>
<tr>
<td>Objectives</td>
<td>Obtain modular computational soundness proofs</td>
</tr>
<tr>
<td>Description</td>
<td>See detailed description below</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deliv.</th>
<th>Ver.</th>
<th>Title</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4.1</td>
<td>Final</td>
<td>Deducibility/indistinguishability games</td>
<td>T0+12</td>
<td>Document</td>
</tr>
<tr>
<td>D4.2</td>
<td>Final</td>
<td>Relevance of games</td>
<td>T0+24</td>
<td>Document</td>
</tr>
<tr>
<td>D4.3</td>
<td>Final</td>
<td>Composability results</td>
<td>T0+36</td>
<td>Document</td>
</tr>
<tr>
<td>D4.4</td>
<td>Final</td>
<td>Application to a computational proof of FOO</td>
<td>T0+48</td>
<td>Document</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validation criteria</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>LORIA</td>
</tr>
<tr>
<td></td>
<td>18 mm.</td>
</tr>
</tbody>
</table>

We plan to develop a framework such that soundness of each primitive can be proved independently, still deriving soundness for the whole set of primitives. The idea is to define abstractly what a good abstraction of a primitive is, independently of the execution model, in the spirit of the definition of adaptive soundness of static equivalence [55]. Compared to the approach of Backes [14], our approach will be somewhat simpler because abstracted from the execution model. Our approach will also avoid the impossibility results for several important primitives [12] due to the universality of Backes library. The intermediate goals will be as follows:

**Task 4.1: Deducibility/indistinguishability games**
The current cryptographic games are not suited to composition. For instance, an IND-CCA game will allow the attacker to query an encryption/decryption oracle, while composed messages include several primitives that may interact. The first task is to design more abstract games, in which the queries can be arbitrary compositions of primitives. These games and the winning conditions will therefore be independent of the actual primitives.

**Task 4.2: Relevance of the games with respect to assumptions**
Our game definition in the previous task will define a new security notion. It is meaningful only if the usual cryptographic assumptions imply the security in the new above sense, when there is a single cryptographic primitive. This is the goal of this task: show that the abstract definition, when specialized to, say, encryption, is implied by some standard security
notions. Typically, IND-CCA2 should be sufficient for public-key encryption (provided that the public key can be extracted from the ciphertext) and INT-CTXT + IND-CPA should be sufficient for symmetric encryption.

Task 4.3: Relevance of the games with respect to soundness proofs
The previous task showed, roughly, that the new notion of security is not too strong, since it is implied by some classical notions. We have then to show that it is strong enough. This task consists in revisiting the new soundness proofs (both the trace mapping and the computational soundness of observational equivalence), recasting them in terms of our games. Succeeding in this task will show that our abstraction is the right intermediate notion.

Task 4.4: Modularity results
The results of the previous tasks should allow us to reduce the general composability problem (our ultimate goal) to the composability of games. This task consists in designing sufficient conditions for the game composability: winning a composed game should imply winning one of the component games. With respect to CRYPTOVERIF and its game transformations, the result of this task should be a general (de)composition result, that does not depend on the particular protocol, and even not on the primitives. With respect to Universal Composability, we expect to get more composability results (for instance exclusive-or should be in the scope of our results), because
1. we do not require a primitive to be secure in any environment, but instead give conditions, under which two functionalities can be safely composed;
2. we only aim at reducing the computational security to the symbolic security: the final security proof will be performed by our symbolic theorem provers.

This is much more practical: automating universal composability proofs seems to be extremely challenging.

Task 4.5: A computational proof of FOO
Finally, we wish to apply our modularity results to several examples of primitives. A good target is the simple e-voting protocol FOO, which involves cryptographic primitives, for which no soundness result is known. It is also a good test for the applicability of the modularity results of the previous section, since we need to consider several cryptographic primitives. Together with the Task 2, the outcome will be an automated proof of FOO, in the computational model (or an attack on the protocol...).

Who does what

<table>
<thead>
<tr>
<th>Task</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>LORIA, LSV</td>
</tr>
<tr>
<td>4.2</td>
<td>LORIA, LSV</td>
</tr>
<tr>
<td>4.3</td>
<td>LORIA, LSV</td>
</tr>
<tr>
<td>4.4</td>
<td>LORIA, LSV</td>
</tr>
<tr>
<td>4.5</td>
<td>LORIA, LSV</td>
</tr>
</tbody>
</table>
4.3.5 TASK 5: LOGIC FOR COMPUTATIONAL INDISTINGUISHABILITY

<table>
<thead>
<tr>
<th>Task 5</th>
<th>Logic for computational indistinguishability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>T0</td>
</tr>
<tr>
<td>End</td>
<td>T0+48</td>
</tr>
<tr>
<td>Duration: 48 months</td>
<td>Task leader: Yassine Lakhnech (VERIMAG)</td>
</tr>
</tbody>
</table>

Objectives
Design and implement a logic for computational indistinguishability

Description
See detailed description below

<table>
<thead>
<tr>
<th>Deliv.</th>
<th>Ver.</th>
<th>Title</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4.1</td>
<td>Initial</td>
<td>First prototype implementation</td>
<td>T0+24</td>
<td>Software</td>
</tr>
<tr>
<td>D4.2</td>
<td>Final</td>
<td>Evaluation of the first prototype implementation on a challenging case study: a fully formalized proof of the IND-CCA security of OAEP</td>
<td>T0+36</td>
<td>Document</td>
</tr>
<tr>
<td>D4.3</td>
<td>Final</td>
<td>Enhanced prototype implementation evaluated on a key exchange protocol</td>
<td>T0+48</td>
<td>Software</td>
</tr>
</tbody>
</table>

Validation criteria
Publications, availability of software

Resources

<table>
<thead>
<tr>
<th>VERIMAG</th>
<th>LSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 mm.</td>
<td>17.8 mm.</td>
</tr>
</tbody>
</table>

We plan to develop a logic for reasoning about indistinguishability of cryptographic systems in the concrete security framework. The concrete security framework allows one to prove bounds on the probability of attacks in dependence of the resources needed to perform the attack. In this framework, one can express that two cryptographic systems are \((t,e)\)-indistinguishable meaning that the probability of any attacker, whose computation time is bounded by \(t\), to distinguish the cryptographic systems is bounded by \(e\). This is a very useful approach since it allows one to compute the required security parameter given the quantitative security goals. Getting this type of analysis seems to be out of the reach for the indirect approach at least using the current approaches. The main objective of the logic is to capture, with as few as possible deduction rules, reasoning steps common in cryptographic proofs. More precisely, the logic will pay particular attention to the interaction with oracles and reasoning steps based on the modification of oracle implementations. In other words, the logic will allow us to ask foundational questions such as completeness of the logic, its decidability and proof search algorithms. We believe that other approaches, for instance those based on games, will benefit from our logic and its formalization.

The intermediate goals of this task will be:

**Task 5.1:** Define a model for cryptographic systems that is general enough to cover complex cryptographic primitives as well as security protocols. The approach we intend to follow here is to view a system as a set of oracles that share a state. An attack is then a sequence of oracle calls. Oracles may be deterministic or probabilistic, stateless or stateful, and may have side effects on the shared state. The composition of an attacker with a system denotes a
distribution on such sequences of oracle calls. It should be possible to embed cryptographic games in this model.

**Task 5.2:** Develop a formal system that allows to reason about such distributions. We need to consider three types of assertions. The first type allows to talk about indistinguishability. The second type is about the probability of events in distributions resulting from the interaction of an attacker with a system of oracles. And the third type of assertions is about distributions identities. It turns out that there is little work on logics that reason about distributions identity, which is mandatory for reasoning about indistinguishability. We will study the completeness of our logic and prove its soundness.

**Task 5.3:** Develop proof search algorithms and heuristics. This is a very challenging task that is mandatory to increase the level of automation of the developed proof system, and hence, to increase the potential of its acceptability. Modularity results obtained in the project and a very close analysis of the soundness results of the symbolic models and their proofs will be of great interest to come up with algorithms and heuristics for proof search.

**Task 5.4:** The model and the formal system should be formalized and implemented on top of a theorem prover (potentially Coq).

**Who does what**

<table>
<thead>
<tr>
<th>Task</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>VERIMAG, LSV</td>
</tr>
<tr>
<td>5.2</td>
<td>VERIMAG, LSV</td>
</tr>
<tr>
<td>5.3</td>
<td>VERIMAG, LSV</td>
</tr>
<tr>
<td>5.4</td>
<td>VERIMAG, LSV</td>
</tr>
</tbody>
</table>

**4.4. TASKS SCHEDULE, DELIVERABLES AND MILESTONES**

**Tasks schedule**

The next figure summarizes the schedule of the various tasks. An arrow from a task A to a task B means that B depends on A. The tasks 2.1 (symbolic model) and 2.2 (computational soundness) will enrich each other. The case study on FOO (task 2.4) and the comparison (task 2.5) depend on the three approaches (2.1, 2.2, and 2.3). The generation of implementations from CryptoVerif specifications (task 3.1) also depends on the extensions of CryptoVerif (task 2.3). The tasks 3.1 and 3.2 depend on each other to guarantee the compatibility between the proofs of primitives in 3.2 and the assumptions made on these primitives in 3.1. The modularity (task 4) and the computational soundness approach (task 2.2) are also mutually dependent. Finally, the computational soundness (2.2) and modularity (4.4) results will be helpful for designing the proof heuristics in task 5.3.
Deliverables and milestones

<table>
<thead>
<tr>
<th>Task number</th>
<th>Date</th>
<th>Title</th>
<th>Person in charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T0+12</td>
<td>Progress report</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>1</td>
<td>T0+24</td>
<td>Progress report</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>1</td>
<td>T0+36</td>
<td>Progress report</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>1</td>
<td>T0+48</td>
<td>Final report</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>2</td>
<td>T0+12</td>
<td>A version of CRYPTOVERIF that handles DDH</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>2</td>
<td>T0+24</td>
<td>Prototype implementation for task 2.1.2</td>
<td>Hubert Comon-Lundh</td>
</tr>
<tr>
<td>2</td>
<td>T0+36</td>
<td>A version of CRYPTOVERIF that handles XOR</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>2</td>
<td>T0+48</td>
<td>Results of our tools on benchmarks</td>
<td>Hubert Comon-Lundh</td>
</tr>
<tr>
<td>3</td>
<td>T0+12</td>
<td>Prototype of compiler from specifications to implementations</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>3</td>
<td>T0+24</td>
<td>Paper on the generation of proved implementations</td>
<td>Bruno Blanchet</td>
</tr>
<tr>
<td>3</td>
<td>T0+24</td>
<td>Paper on the verification of cryptographic primitives</td>
<td>Yassine Lakhnech</td>
</tr>
<tr>
<td>3</td>
<td>T0+48</td>
<td>Enhanced compiler from specifications to implementations</td>
<td>Bruno Blanchet</td>
</tr>
</tbody>
</table>
Threats

The main threat for this project is not to obtain no result, but the obtained results may have a more limited scope than we hope or may require more time than expected. For example, our tools may fail to prove particular protocols, although they support the primitives used in these protocols.

Meetings

The project meetings will be organized every six months, alternatively at each partner location or joint with the FCC workshop (see Section 4.2).

5. Dissemination and exploitation of results. Management of intellectual property

We will of course publish our work in international conferences and journals; for instance, the conferences CSF, IEEE Symposium on Security and Privacy, ESORICS, ICALP, or also CRYPTO for the cryptographic aspects, would be good targets. The Computer Security Foundations Symposium (CSF), in particular, will be organised by LSV in 2011. We also plan to continue our involvement in the workshop on Formal and Computational Cryptography (FCC), which constitutes an ideal forum for more informal discussions, specialized on the topic of this project. The project will also produce software, which we plan to make available on the web under a free licence, such as CeCILL, so that other teams can benefit from our work. We believe that this is the best way to make our work visible and useful, since the target users for our tools will, in a first stage, be other researchers that design or verify protocols. We will protect our software by depositing it at APP (Agence pour la Protection des Programmes).

As far as communication to the public is concerned, we will make popularisation efforts. In particular, we will submit a paper to Rue89 (a French news web site).

Since this is a basic research project, it will have an important scientific and technical impact, with a better understanding of the various protocol models and their relations, and practical ways to get stronger guarantees on security protocols. However, it will also have important
and immediate practical consequences: we will produce proofs or discover flaws and propose fixes in protocols that are currently used in practice. We will also produce tools that can be used by other teams that design or verify protocols. We believe that, in the very short term, the work done in this project will have a strong influence on semi-industrial groups, such as DGA or Microsoft Research, where some of our previous tools (for instance, PROVERIF and CRYPTOVERIF) have already been used to verify Microsoft products. In a longer term, we hope that the techniques and tools we will develop get widely used in industry for verifying protocols, and we will certainly do our best for that, but this is a slow evolution.

6. CONSORTIUM DESCRIPTION

6.1. PARTNERS DESCRIPTION & RELEVANCE, COMPLEMENTARITY

INRIA Paris-Rocquencourt

The project-team CASCADE of INRIA Paris-Rocquencourt has a strong expertise in cryptography, in particular in manual security proofs in the computational model (David Pointcheval and his team). It also has a strong expertise in formal methods for the automatic verification of security protocols: B. Blanchet is the main author of the protocol verifiers PROVERIF and CRYPTOVERIF. In particular, CRYPTOVERIF is the first automated protocol verifier that works directly in the computational model and produces proofs by sequences of games, as cryptographers do. Moreover, B. Blanchet also has expertise in program analysis, which will be useful for the verification of implementations of protocols. All these aspects will be important for the success of this project.

Laboratoire Spécification et Vérification (LSV, ENS Cachan)

The LSV Partner is a laboratory that is specialized in verification. This covers model-checking techniques of infinite or timed systems, and also verification of security, through the Sécurité des Systèmes d’Information (SECSI) project. For instance, we have proposed specifications of security properties and developed exact techniques for the decision of security in the case of a bounded number of sessions, for various cryptographic primitives. We also designed proof methods using upper approximations based on tree automata (the H1 tool). We also have expertise in security proofs in the cryptographic model, for instance through collaborations with the LORIA group.

Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA)

The LORIA partner has a strong expertise in showing how symbolic models can be sound abstraction of cryptographic ones. With VERIMAG, it was the very first team in France to show that security proofs in symbolic models can directly imply security proofs in cryptographic models, under classical assumptions on the primitives. Relying on the tool platform AVISPA developed at LORIA (Cassis group), we have proposed one of the first tools (with CRYPTOVERIF, developed by the INRIA Paris-Rocquencourt partner) that enables
automatic security proofs in cryptographic models. Moreover, the LORIA partner has also a good expertise on developing composition results in symbolic models [40] and is currently investigating composition results in collaboration with Bogdan Warinschi (University of Bristol, UK).

VERIMAG

Verimag is a leading laboratory in automated verification, model-checking and abstract interpretation. It has also expertise in theorem-proving and its combination with model-checking and abstraction-based verification. The DCS-team, involved in this project, has been working on the verification of security protocols since almost ten years. The team has proposed and implemented an abstraction-based verification method for security protocols, developed soundness results for the symbolic model for protocols that combine or include asymmetric and symmetric encryption, digital signature and hash function. The team has also developed and implemented automated analyses for verifying implementations of cryptographic primitives: asymmetric encryption and symmetric encryption modes.

Adequacy of the partnership

We believe that our group of four small teams is one of the best configurations for achieving the goals of the project (and the best group in France for achieving the goal). We all worked in the past on both symbolic and computational proofs of security, which is the main research topic of all of us. We bring together in this project researchers with backgrounds in formal methods and computational cryptography; combining their knowledge in these topics will be key to the success of the project. We have also worked on several different approaches (computational soundness proofs, direct proofs in the computational model using logics or automated tools), which are complementary. Each approach has its advantages and limitations; considering several of them, comparing them, and combining their best aspects is one of the goals of this project.

We also have experience of past successful collaborations in several projects, including the ANR projects FORMACRYPT (LIENS, LSV, LORIA) and AVOTE (LSV, LORIA, VERIMAG), the RNTL projects PROUVÉ (CRIL Technology Systèmes Avancés, France Télécom R&D, INRIA Lorraine, LSV, VERIMAG) and EVA (LSV, VERIMAG, Trusted Logic), and the ACI ROSSIGNOL (LIF, LIX, LSV, VERIMAG) and VERNAM (LSV, Université de Provence, LORIA). As we are not involved significantly in other research projects, we will devote our energy to this project.

6.2. RELEVANT EXPERIENCE OF THE PROJECT COORDINATOR

Bruno Blanchet already coordinated the previous FORMACRYPT project between LIENS, LSV, and LORIA funded by ANR (January 2006-July 2009). This project was very successful: it produced 6 papers in international journals, 3 invited talks in international conferences, 19 papers in international conferences, 11 workshop presentations, 3 PhD theses, 1 habilitation to supervise research. It has also lead to two tools: a module in the protocol verifier AVISPA
and CRYPTOVERIF, to a series of workshops on Formal and Computational Cryptography (FCC) and to a Spring School and French-Japanese collaboration workshop on Computational and Symbolic Proofs of Security (CoSyProofs’09). B. Blanchet is also a well-known expert in the field of automatic security protocol verification, and the main author of the protocol verifiers PROVERIF and CRYPTOVERIF. He is the author of 10 papers in international journals, 3 invited talks in international conferences, 19 papers in international conferences. He was member of 14 programme committees of international conferences, including CSF(W), POPL, PLDI, ESOP, TACAS, FOSSACS, CONCUR. He participated to several previous research projects, including the European project Daedalus, the Action Spécifique Sécurité from CNRS, and the RNTL project Astrée.

7. **Scientific Justification for the Mobilisation of the Resources**

7.1. **Partner 1: INRIA Paris-Rocquencourt**

- **Equipment**
  
  Nothing.

- **Personnel costs**
  
  A master student on task 2.  
  5 months  
  8000€

  1 PhD grant, shared with LSV, LORIA, or VERIMAG.  
  18 months  
  53397€

  (We request 2 PhD grants for the whole project. Since these grants are shared between several laboratories, we count half a grant for each laboratory and group the profiles of these positions in Section 7.5 below.)

  1 post-doc.  
  12 months  
  55844€

  The goal of this post-doc will be to develop computational soundness results (task 2.2), in particular to weaken security assumptions of the current computational soundness results, to get more widely applicable results.

- **Subcontracting**
  
  Nothing.

- **Travel**

  **Project meetings:**

  Project meetings in Nancy  
  2 meetings  
  300€/pers./meeting  
  4 pers.  
  2400€

  Project meetings in Grenoble  
  2 meetings  
  300€/pers./meeting  
  4 pers.  
  2400€

  Project meeting joint with CSF/FCC  
  1 meeting  
  300€/pers.  
  4 pers.  
  1200€

  **Travel expenses to conferences for dissemination of the results:**

  Conferences in France  
  4 conf.  
  500€/pers./conf.  
  1 pers.  
  2000€

  Conferences in foreign countries  
  16 conf.  
  2000€/pers./conf.  
  1 pers.  
  32000€

  **Total**  
  40000€
• *Expenses for inward billing (Costs justified by internal procedures of invoicing)*
  Nothing.

• **Other working costs**
  2 desktop computers 1500€ each 3000€
  2 notebook computers 2500€ each 5000€
  **Total** 8000€

These computers are requested to renew the desktop computers and notebooks of the participants of the project.

### 7.2. PARTNER 2: LABORATOIRE SPECIFICATION ET VERIFICATION (LSV)

• **Equipment**
  Nothing.

• **Personnel costs**
  A master student on task 5. 5 months 8000€
  1 PhD grant, shared with INRIA Paris-Rocquencourt or LORIA. 18 months 53397€
  (We request 2 PhD grants for the whole project. Since these grants are shared between several laboratories, we count half a grant for each laboratory and group the profiles of these positions in Section 7.5 below.)

• **Subcontracting**
  Nothing.

• **Travel**
  **Project meetings:**
  Project meetings in Nancy 2 meetings 300€/pers./meeting 4 pers. 2400€
  Project meetings in Grenoble 2 meetings 300€/pers./meeting 4 pers. 2400€
  Project meeting joint with CSF/FCC 1 meeting 300€/pers. 4 pers. 1200€

  **Travel expenses to conferences for dissemination of the results:**
  Conferences in France 4 conf. 500€/pers./conf. 1 pers. 2000€
  Conferences in foreign countries 16 conf. 2000€/pers./conf. 1 pers. 32000€
  **Total** 40000€

• *Expenses for inward billing (Costs justified by internal procedures of invoicing)*
  Nothing.

• **Other working costs**
  2 desktop computers 1500€ each 3000€
  2 notebook computers 2500€ each 5000€
  **Total** 8000€

These computers are requested to renew the desktop computers and notebooks of the participants of the project.
7.3. PARTNER 3: LABORATOIRE LORRAIN DE RECHERCHE EN INFORMATIQUE ET SES APPLICATIONS (LORIA)

- **Equipment**
  Nothing.

- **Personnel costs**
  A master student on task 2. 5 months  8000€
  A master student on task 4. 5 months  8000€
  1 PhD grant, shared with LSV or INRIA Paris-Rocquencourt. 18 months  53397€
  (We request 2 PhD grants for the whole project. Since these grants are shared between several laboratories, we count half a grant for each laboratory and group the profiles of these positions in Section 7.5 below.)

- **Subcontracting**
  Nothing.

- **Travel**
  **Project meetings:**
  Project meetings in Paris  3 meetings  300€/pers./meeting  2 pers.  1800€
  Project meeting in Cachan  1 meeting  300€/pers.  2 pers.  600€
  Project meetings in Grenoble  2 meetings  300€/pers./meeting  2 pers.  2400€
  Project meeting joint with CSF/FCC  1 meeting  300€/pers.  2 pers.  1200€
  **Travel expenses to conferences for dissemination of the results:**
  Conferences in France  4 conf.  500€/pers./conf.  1 pers.  2000€
  Conferences in foreign countries  11 conf.  2000€/pers./conf.  1 pers.  22000€
  **Total**  30000€

- **Expenses for inward billing (Costs justified by internal procedures of invoicing)**
  Nothing.

- **Other working costs**
  2 desktop computers  1500€ each  3000€
  2 notebook computers  2500€ each  5000€
  **Total**  8000€
  These computers are requested to renew the desktop computers and notebooks of the participants of the project.

7.4. PARTNER 4: VERIMAG

- **Equipment**
  Nothing.

- **Personnel costs**
  A master student on task 5. 5 months  8000€
1 PhD grant, shared with INRIA Paris-Rocquencourt. 18 months 53397€
(We request 2 PhD grants for the whole project. Since these grants are shared between several laboratories, we count half a grant for each laboratory and group the profiles of these positions in Section 7.5 below.)
1 post-doc. 12 months 55844€
The goal of this post-doc will be to work on the correctness of implementations of cryptographic primitives (task 3.2) and on proof search algorithms for the indistinguishability logic (tasks 5.3 and 5.4).

- **Subcontracting**
  Nothing.

- **Travel**
  **Project meetings:**
  - Project meetings in Paris 3 meetings 300€/pers./meeting 5 pers. 4500€
  - Project meeting in Cachan 1 meeting 300€/pers. 5 pers. 1500€
  - Project meetings in Nancy 2 meetings 300€/pers./meeting 5 pers. 3000€
  - Project meeting joint with CSF/FCC 1 meeting 300€/pers. 5 pers. 1500€
  **Travel expenses to conferences for dissemination of the results:**
  - Conferences in France 3 conf. 500€/pers./conf. 1 pers. 1500€
  - Conferences in foreign countries 14 conf. 2000€/pers./conf. 1 pers. 28000€
  **Total** 40000€

- **Expenses for inward billing (Costs justified by internal procedures of invoicing)**
  Nothing.

- **Other working costs**
  - 2 desktop computers 1500€ each 3000€
  - 2 notebook computers 2500€ each 5000€
  **Total** 8000€
The computers are requested to renew the desktop computers and notebooks of the participants of the project.

### 7.5. Profiles for PhD positions

**Profile 1:**

The goal of this PhD is to develop a theorem prover (and its theoretical foundations), that corresponds to the Task 2.1.2 and, in some respects, to the Task 2.2. More precisely, the PhD candidate should develop algorithms for deciding the symbolic equivalence of processes, stepwise enlarging the scope of the prover to large classes of protocols/cryptographic primitives. One of the goals will be to be able, by the end of the thesis, to prove (or disprove) automatically indistinguishability properties for an e-voting protocol.
This thesis can be a joint LSV-INRIA-Paris-Rocquencourt thesis. A financial support has already been requested to the region Ile de France. In case we get the support from the region, the ANR support will be used to recruit a student on another profile.

Profile 2:

The goal of this PhD roughly corresponds to the Task 2.2. The first step is the Task 2.2.1: design a formal model and a computational soundness result that would account for dishonest key generation. If this can be completed within a reasonable time, the other parts of the task can be addressed by the student, typically computational soundness results for other cryptographic primitives.

This thesis can be a joint LORIA-LSV thesis. There is no other financial support requested so far.

Profile 3:

The goal of this PhD is to extend symbolic verification to group protocols (Task 2.1.1). More precisely, the goal is to extend the symbolic verifier PROVERIF developed by INRIA Paris-Rocquencourt so that it can verify protocols with a parametric number of participants. Such protocols typically contain messages whose form depends on the number of participants. This work involves both extensions of the resolution algorithm on Horn clauses used by PROVERIF and of the translation from the protocol specification into Horn clauses. It also requires the treatment of primitives modelled by complex equational theories, such as Diffie-Hellman key agreements and exclusive or, which are often used by group protocols. This work builds upon previous work on group protocols by Steve Kremer, Antoine Mercier, and Ralf Treinen at LSV and by Najah Chridi, Mathieu Turuani, and Michael Rusinowitch.

This thesis can be a joint INRIA Paris-Rocquencourt-LSV or INRIA Paris-Rocquencourt-LORIA thesis. A financial support has already been requested (INRIA CORDI-S grant). In case we get that grant, the ANR support will be used to recruit a student on another profile.

Profile 4:

The goal of this PhD is to develop a logic for reasoning about cryptographic systems based on indistinguishability and probabilities of events (Task 5) and to relate it to other approaches for the same goal, in particular the tool CRYPTOVERIF developed at INRIA Paris-Rocquencourt (Task 2.3). Game-based methods provide a notation for describing interaction between adversaries and cryptographic systems. In this setting, a proof is represented as a sequence of games. The transformation of a game to the next one is often justified in an informal way. The logic to be developed in this Ph.D. work should provide a limited set of rules for reasoning about oracles in presence of an adaptive adversary. The logic should be proved sound and its applicability demonstrated on substantial case studies. It is also
intended to be the basis for developing, or at least extending existing, tool support for proving correctness of cryptographic systems, such as the tool CRYPTOVERIF.

This thesis can be joint INRIA Paris-Rocquencourt-Verimag. There is no other financial support requested.

8. ANNEXES

8.1. REFERENCES


