Soundness in presence of active adversaries

Pascal Lafourcade
VERIMAG, CNRS, University of Grenoble

Summary Security protocols are short programs that aim at securing communication over a public network. Their design is known to be error-prone with flaws found years later. That is why they deserve a careful security analysis, with rigorous proofs. Two main lines of research have been (independently) developed to analyze the security of protocols. On the one hand, formal methods provide with symbolic models and often automatic proofs. On the other hand, cryptographic models propose a tighter modeling but proofs are more difficult to write and to check. There are two competing approaches to the verification of cryptographic protocols. In the so-called formal (also called Dolev-Yao) model, data are specified using abstract data types (algebraic specification) and are manipulated by honest agents and adversaries according to the operations of the abstract data types. In other words, the abstract data types give an abstract specification of the cryptographic primitives and of the computational power of the adversaries that try to break the security properties. The verification techniques discussed in the previous tasks are based on this model. On the other hand, in the complexity-theoretic model, also called the computational model, the adversary can be any polynomial-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 polynomial-time probabilistic algorithm. Moreover, in this model security properties are expressed in terms of the probability of success of any polynomial-time attack. Attacks are usually defined in terms of probabilistic games, where the adversary has access to some oracles and wins the game, if 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 an approach, developed during the last decade, consists in bridging the two approaches, showing that symbolic models are sound w.r.t. symbolic ones, yielding strong security guarantees using automatic tools, in order to get the best of both worlds. These results have been developed for several cryptographic primitives (e.g. symmetric and asymmetric encryption, signatures, hash) and security properties.

In the previous deliverable D3.1, we have presented a survey of the existing results on the computational soundness of symbolic indistinguishability relations in the presence of a passive adversary, for which several results were obtained by the members of the AVOTÉ project.

In this report, we look at the soundness in presence of active adversaries. As stated earlier the soundness of the Dolev-Yao model for protocols that use asymmetric encryption, symmetric encryption, signature and hash functions has been studied. More precisely, it has been established that for trace properties under suitable assumptions about these primitives the non-existence of a symbolic attack implies that the probability of polynomial-time attacks in the computational model is negligible. This report summarize the following papers:

- Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08), Alexandria, Virginia, USA, October 2008. ACM Press [CLC08]. **See Section 1.1.**

- Hubert Comon-Lundh and Véronique Cortier. How to prove security of communication protocols? a discussion on the soundness of formal models w.r.t. computational ones. In Proceedings of the 28th Annual Symposium on Theoretical Aspects of Computer Science

(STACS'11), volume 9 of Leibniz International Proceedings in Informatics, 2011 [CC11]. **See Section 1.2.**

- Judicaël Courant, Marion Daubignard, Pascal Lafourcade Cristian Ene, and Yassine Lahknech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In Proc. 15th ACM Conference on Computer and Communications Security, (CCS'08), Alexandria, USA, 2008. [CDCEL08, CDE+11]. **See Section 2.1.**
- Yassine Lakhnech Martin Gagné, Pascal Lafourcade and Reihaneh Safavi-Naini. Automated proofs for encryption modes. In 13th Annual Asian Computing Science Conference Focusing on Information Security and Privacy: Theory and Practice (ASIAN'09), 2009 [MGSN09]. **See Section 2.2.**
- Cristian Ene, Yassine Lakhnech, and Van Chan Ngo. Formal indistinguishability extended to the random oracle model. ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, 2009 [ELN09]. **See Section 2.3.**

For each of these results, the corresponding publication is appended to this report.

1 Computational Soundness

1.1 Computational Soundness of Observational Equivalence [CLC08]

Many security properties are naturally expressed as indistinguishability between two versions of a protocol. We show that computational proofs of indistinguishability can be considerably simplified, for a class of processes that covers most existing protocols. More precisely, we show a soundness theorem, following the line of research launched by Abadi and Rogaway in 2000: computational indistinguishability in presence of an active attacker is implied by the observational equivalence of the corresponding symbolic processes. We prove our result for symmetric encryption. There is a well-known similar notion in concurrency theory: observational equivalence, introduced by Milner and Hoare in the early 80s. Two processes P and Q are observationally equivalent, denoted by $P \sim_O Q$, if for any process O (a symbolic observer) the processes $P||O$ and $Q||O$ are equally able to emit on a given channel. This means that O cannot observe any difference between P and Q. Observational equivalence is therefore a natural candidate for the symbolic counterpart of indistinguishability, the attacker being replaced by the observer. And indeed, we show a result of the form "two networks of machines are indistinguishable if the processes they implement are observationally equivalent". As a consequence, proving computational indistinguishability can be reduced to proving observational equivalence (for a class of protocols and assuming some standard security hypotheses on the cryptographic primitives). This is a simpler task, which can be completely formalized and sometimes automated as for instance in Proverif tool. We prove our result for symmetric encryption and pairing, using a fragment of the applied pi-calculus for specifying protocols and relying on standard cryptographic assumptions (IND-CPA and INT-CTXT) as well as hypotheses. The fragment of applied pi-calculus we consider allows to express an arbitrary (unbounded) number of sessions of a protocol. The main technical ingredient of our proof is the introduction of tree soundness in the case of passive attackers and the use of intermediate structures, which we called computation trees: on one end such trees roughly correspond to the labeled transition semantics of some process algebra, and, on the other end, they are interpreted as an encryption oracle, scheduled by the attacker. These notions are defined independently of the cryptographic setting. Tree soundness captures the fact that even a passive attacker can adaptively choose its requests. We can then derive a general method for proving that observational equivalence implies computational indistinguishability.

1.2 A Discussion on the Soundness of Formal Models [CC11]

While proving soundness of symbolic models is a very promising approach, several technical details are often not satisfactory. We focus on symmetric encryption and we describe the difficulties

and limitations of the available results. Among all the limitations we will discuss, the main one is to consider only honestly generated keys (or a certifying infrastructure), which is completely unrealistic. There are (at least) two main ways to overcome this assumption. A first possibility, consists in enriching the symbolic model by letting the adversary create new symbolic equalities when building new (dishonest) keys. In this way, many protocols should still be provably secure under the IND-CCA assumption, yet benefiting from a symbolic setting for writing the proof. A second option is to seek for stronger security assumptions by further requesting non-malleability. The idea is that a ciphertext should not be opened to a different plaintext, even when using dishonest keys. This could be achieved by adding a commitment to the encryption scheme. However all these limitations also demonstrate that it is difficult to make symbolic and computational models coincide. Even for standard security primitives, soundness results are very strong since they provide with a generic security proof for any possible protocol (contrary to CryptoVerif). For primitives with many algebraic properties like Exclusive Or or modular exponentiation, the gap between symbolic and computation models is even larger and would require a lot of efforts. We still believe that computational proofs could benefit from the simplicity of symbolic models, yielding automated proofs.

An alternative approach to soundness results could consist in developing directly a logic for proving automatically the results in computational world. It is the approach followed by members of AVOTÉ project that is presented in the next Section.

2 Logic Approach

2.1 Hoare Logic for proving encryption schemes [CDCEL08, CDE$^+$11]

Many generic constructions for building secure cryptosystems from primitives with lower level of security have been proposed. Providing security proofs has also become standard practice. There is, however, a lack of automated verification procedures that analyze such cryptosystems and provide security proofs. In [CDCEL08, CDE$^+$11], we present an automated proof method for analyzing generic asymmetric encryption schemes in the random oracle model (ROM). Generic encryption schemes aim at transforming schemes with weak security properties, such as onewayness, into schemes with stronger security properties, specifically security against chosen ciphertext attacks. We propose a compositional Hoare logic for proving IND-CPA security. An important feature of our method is that it is not based on a global reasoning as it is the case for the game-based approach. Instead, it is based on local reasoning. Indeed, both approaches can be considered complementary as the Hoare logic-based one can be considered as aiming at characterizing by means of predicates the set of contexts in which the game transformations can be applied safely.

Moreover in [CDCEL08] we present a simple criterion for plaintext awareness (PA). Plaintext awareness has been introduced by Bellare and Rogaway. It has then been refined later such that if an encryption scheme is PA and IND-CPA then it is IND-CCA. Intuitively, PA ensures that an adversary cannot generate a valid cipher without knowing the plaintext, and hence, the decryption oracle is useless for him. The definition of PA is complex and proofs of PA are also often complex. We present a simple syntactic criterion that implies plaintext awareness. Roughly speaking the criterion states that the cipher should contain as a sub-string the hash of a bitstring that contains as substrings the plaintext and the random seed. This criterion applies for many schemes and easy to check. Although (or maybe because) the criterion is simple, the proof of its correctness is complex. Putting together these two results, we get a proof method for IND-CCA security.

2.2 Hoare Logic for Encryption Modes [MGSN09]

In the same line of work as the previous works, we also propose a compositional Hoare logic for proving semantic security of modes of operation for symmetric key block ciphers. We notice that many modes use a small set of operations such as xor, concatenation, and selection of random values. We introduce a simple programming language to specify encryption modes and an assertion

language that allows to state invariants and axioms and rules to establish such invariants. The assertion language requires only four predicates: one that allows us to express that the value of a variable is indistinguishable from a random value when given the values of a set of variables, one that states that an expression has not been yet submitted to the block cipher, and two bookkeeping predicates that allow us to keep track of "fresh" random values and counters. Transforming the Hoare logic into an (incomplete) automated verification procedure is quite standard. Indeed, we can interpret the logic as a set of rules that tell us how to propagate the invariants backwards. Using our method, an automated prover could verify semantic security of several encryption modes including CBC, CFB, CTR and OFB. Of course our system does not prove ECB mode, because ECB is not semantically secure.

2.3 Formal indistinguishability extended to the ROM [ELN09]

Several generic constructions for transforming one-way functions to asymmetric encryption schemes have been proposed. One-way functions only guarantee the weak secrecy of their arguments. That is, given the image by a one-way function of a random value, an adversary has only negligible probability to compute this random value. Encryption schemes must guarantee a stronger secrecy notion. They must be at least resistant against indistinguishability-attacks under chosen plain-text text (IND-CPA). Most practical constructions have been proved in the random oracle model. Such computational proofs turn out to be complex and error prone. We would like to be able to treat generic encryption schemes that transform one-way functions to IND-CPA secure encryption schemes. Bana et al. in [BM$^+$06] have introduced Formal Indistinguishability Relations, (FIR for short) as an appropriate abstraction of computational indistinguishability. We extend Bana et al.'s approach by introducing a notion of symbolic equivalence that allows us to prove security of encryption schemes symbolically. Therefore, three problems need to be solved. First, we need to cope with one-way functions. This is another example where static equivalence does not seem to be appropriate. The second problem that needs to be solved is related to the fact that almost all practical provably secure encryption schemes are analyzed in the random oracle model. Thus, we need to be able to symbolically prove that a value of a given expression a cannot be computed by any adversary. To cope with this problem, our notion of symbolic indistinguishability comes along with a non-derivability symbolic relation. Thus in our approach, we start from an initial pair of a non-derivability relation and a frame equivalence relation. Then, we provide rules that define a closure of this pair of relations in the spirit of Bana et al.'s work. Also in our case, soundness of the obtained relations can be checked by checking soundness of the initial relations. The third problem is related to the fact that security notions for encryption schemes such IND-CPA and real-or-random indistinguishability of cipher-text under chosen plaintext involve a generated from of active adversaries. Indeed, these security definitions correspond to two-phase games, where the adversary first computes a value, then a challenge is produced, the the adversary tries to solve the challenge. Static equivalence and FIR (as defined in [BM$^+$06]) consider only passive adversaries. To solve this problem we consider frames that include variables that correspond to adversaries. As frames are finite terms, we only have finitely many such variables. This is the reason why we only have a degenerate form of active adversaries which is enough to treat security of encryption schemes and digital signature, for instance. We illustrate the framework by considering security proofs of the construction of Bellare and Rogaway and Hash El Gamal.

References

[BM$^+$06] Gergei Bana, Payman Mohassel, , Till Stegers, and Till Stegers. Computational soundness of formal indistinguishability and static equivalence. In *In Proc. 11th Asian Computing Science Conference (ASIAN'06), LNCS*. Springer, 2006.

[CC11] Hubert Comon-Lundh and Véronique Cortier. How to prove security of communication protocols? a discussion on the soundness of formal models w.r.t. computational ones. In Christoph Dürr and Thomas Schwentick, editors, *Proceedings of the 28th*

Annual Symposium on Theoretical Aspects of Computer Science (STACS'11), volume 9 of *Leibniz International Proceedings in Informatics*, pages 29–44, Dortmund, Germany, March 2011. Leibniz-Zentrum für Informatik.

[CDCEL08] Judicaël Courant, Marion Daubignard, Pascal Lafourcade Cristian Ene, and Yassine Lahknech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In *Proc. 15th ACM Conference on Computer and Communications Security, (CCS'08)*, Alexandria, USA, 2008. To appear.

[CDE+11] Judicaël Courant, Marion Daubignard, Cristian Ene, Pascal Lafourcade, and Yassine Lakhnech. Automated proofs for asymmetric encryption. *J. Autom. Reasoning*, 46(3-4):261–291, 2011.

[CLC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, Alexandria, Virginia, USA, October 2008. ACM Press.

[ELN09] Cristian Ene, Yassine Lakhnech, and Van Chan Ngo. Formal indistinguishability extended to the random oracle model. In Michael Backes and Peng Ning, editors, *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*, volume 5789 of *Lecture Notes in Computer Science*, pages 555–570. Springer, 2009.

[MGSN09] Yassine Lakhnech Martin Gagné, Pascal Lafourcade and Reihaneh Safavi-Naini. Automated proofs for encryption modes. In *13th Annual Asian Computing Science Conference Focusing on Information Security and Privacy: Theory and Practice (ASIAN'09)*, Urumqi, China, oct 2009.

Computational Soundness of Observational Equivalence

Hubert Comon-Lundh
Research Center for Information Security and
ENS Cachan
AIST, Akihabara-Daibiru, Tokyo, Japan
h.comon-lundh@aist.go.jp

Véronique Cortier*
LORIA, CNRS & INRIA project CASSIS
Nancy, France
cortier@loria.fr

ABSTRACT

Many security properties are naturally expressed as indistinguishability between two versions of a protocol. In this paper, we show that computational proofs of indistinguishability can be considerably simplified, for a class of processes that covers most existing protocols. More precisely, we show a soundness theorem, following the line of research launched by Abadi and Rogaway in 2000: computational indistinguishability in presence of an active attacker is implied by the *observational equivalence* of the corresponding symbolic processes.
We prove our result for symmetric encryption, but the same techniques can be applied to other security primitives such as signatures and public-key encryption. The proof requires the introduction of new concepts, which are general and can be reused in other settings.

Categories and Subject Descriptors

D.2.4 [**Verification**]: Formal methods

General Terms

Verification

1. INTRODUCTION

Two families of models have been designed for the rigorous analysis of security protocols: the so-called Dolev-Yao (symbolic, formal) models on the one hand and the cryptographic (computational, concrete) models on the other hand. In symbolic models messages are formal terms and the adversary can only perform a fixed set of operations on them. The main advantage of the symbolic approach is its relative simplicity which makes it amenable to automated analysis tools [14] In cryptographic models, messages are bit strings and the adversary is an arbitrary probabilistic polynomial-time (ppt) Turing machine. While the proofs in such models yield strong security guarantees, they are often quite involved and seldom suitable for automation.

Starting with the seminal work of Abadi and Rogaway [4], a lot of efforts has been directed to bridging the gap between the two approaches. The goal is to obtain the best of both worlds: simple, automated security proofs that entail strong security guarantees. The numerous relevant works can be divided into two categories. In the first one ([1, 12, 31] and many others), the authors generalize Abadi and Rogaway result, typically considering a larger set of security primitives. However, they still only consider a *passive adversary*. This rules out the so-called "man-in-the-middle attacks". Analyzing real protocols requires to consider active adversaries, which is the aim of the second category of papers (e.g. [8, 18, 22, 30]). It is also the aim of the present paper. We consider however a wider class of security properties.

Trace properties vs. Equivalence properties. We call here a *trace property* a formal statement that something bad never occurs on any trace of a protocol. (Formally, this is a property definable in linear time temporal logic). Integrity and authentication are examples of trace properties. That is why they were the first for which computational guarantees were derived out of symbolic ones [10, 32].
There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties.

- Anonymity states that any *two* execution traces, in which names are swapped, cannot be distinguished by an attacker. More precisely, anonymity requires two instances of the protocol P_{AB} and P_{BA}, the names A, B being switched in the second copy. An adversary interacting with one of the two copies should not be able to tell (with non-negligible probability) with which copy he is interacting. There is no known way to reduce this problem to a property of a single protocol copy.
  Privacy related properties involved in electronic voting protocols [23] also use an equivalence and cannot be expressed in linear temporal logic.

- Similarly, in the computational worlds, anonymity of group signatures [5] is defined through the indistinguishability of two games where different identities are used in each game. A similar definition is proposed for "blindness" of signatures in [27].

- The "computational secrecy" states that the protocol does not leak any piece of the secret (this is sometimes

*This work has been partially supported by the ARA SSIA FormaCrypt and the ARA project AVOTÉ.

CCS'08 October 27–31, 2008, Alexandria, Virginia, USA.

called "real or random"). Such a property is naturally expressed as an indistinguishability property: the attacker cannot distinguish between two games, one of which is the real protocol, and, in the other one, the secret has been replaced by a random string. There are several works [32, 9, 22, 26, 18, 21]showing how to soundly abstract it as a trace property in the symbolic model, in a number of particular cases. It is not clear, however, that such a property can be expressed as a trace property in general. Consider e.g. the case of a hash function and assume that a protocol reveals the hash $h(s)$ of some secret s. Then s cannot be computed (by one-wayness of h), which, from the trace property point of view, would be sufficient for confidentiality. On the other hand, an attacker certainly learns something about s and the computational secrecy is not preserved.

- Strong (also called "black-box") simulatability [11, 29], states that, given a real protocol P and an ideal functionality F, there is a simulator S such that P cannot be distinguished from $S\|F$ by any environment. Again, this is not a property of any particular trace, but rather a relationship between the real traces and the ideal ones. Various notions of *universal composability* [17, 19] can be defined in a similar way.

This shows the importance and generality of indistinguishability properties compared to trace properties.

The main question is then: "is it possible to get sound abstraction results for computational indistinguishability, analogous to the results obtained so far for trace properties ?" This is the question, which we want to address in this paper, for a sample set of cryptographic primitives.

Our contribution. There is a well-known similar notion in concurrency theory: observational equivalence, introduced by Milner and Hoare in the early 80s. Two processes P and Q are observationally equivalent, denoted by $P \sim_o Q$, if for any process O (a symbolic observer) the processes $P\|O$ and $Q\|O$ are equally able to emit on a given channel. This means that O cannot observe any difference between P and Q. Observational equivalence is therefore a natural candidate for the symbolic counterpart of indistinguishability, the attacker being replaced by the observer. And indeed, we show in this paper a result of the form "two networks of machines are indistinguishable if the processes they implement are observationally equivalent". As a consequence, proving computational indistinguishability can be reduced to proving observational equivalence (for a class of protocols and assuming some standard security hypotheses on the cryptographic primitives). This is a simpler task, which can be completely formalized and sometimes automated [15, 24].

We prove our result for symmetric encryption and pairing, using a fragment of the applied pi-calculus [2] for specifying protocols and relying on standard cryptographic assumptions (IND-CPA and INT-CTXT) as well as hypotheses, which are similar to those of [8]. The main difference with this latter work is that we prove the soundness of observational equivalence, which covers more properties. The fragment of applied pi-calculus we consider allows to express an arbitrary (unbounded) number of sessions of a protocol.

To prove our result, we need first to show that any computational trace is, with overwhelming probability, an instance of a symbolic one. This lemma is similar to [22, 26], though with different hypotheses and in a different model. A naive idea would be then to consider any pair of related symbolic traces: by observational equivalence (and actually labeled bisimilarity) the two traces are statically equivalent. Then we could try to apply soundness of static equivalence on these traces (using results in the passive case, e.g. [4, 1, 12, 31]). This idea does not work, since the computational traces could be spread over the symbolic ones: if there is only one computational trace corresponding to a given symbolic trace, then the symbolic traces indistinguishability does not tell us anything relevant on the computational ones.

That is why we need a new tool; the main technical ingredient of our proof is the introduction of *tree soundness* in the case of passive attackers and the use of intermediate structures, which we called *computation trees*: on one end such trees roughly correspond to the labeled transition semantics of some process algebra, and, on the other end, they are interpreted as an encryption oracle, scheduled by the attacker. These notions are defined independently of the cryptographic setting. Tree soundness captures the fact that even a passive attacker can *adaptively* choose its requests. It seems related to "adaptive soundness of static equivalence" as defined in [28] though no precise relationship has been established yet. We can then derive a general method for proving that observational equivalence implies computational indistinguishability. We believe our techniques are general and can be reused in other settings. In particular, using our generic approach, it should not be difficult to extend our result to other security primitives like asymmetric encryption and signatures.

Related Work. In a series of papers starting with Micciancio and Warinschi [32] and continued with e.g. [22, 26], the authors show trace mapping properties: for some selected primitives (public-key encryption and signatures in the above-cited papers) they show that a computational trace is an instance of a symbolic trace, with overwhelming probability. We have a similar result for symmetric encryption in the present paper, but this is not our main contribution.

There is a huge amount of work on simulatability/universal composability, especially the work of Backes *et. al.* and Canetti [17, 11, 10, 8, 9]. In the black-box simulatability approach of [11], which we refer to as BPW, the symbolic model is different than ours: there are essential constructions such as handles, which we do not have in our (more abstract) symbolic model, that is a standard process algebra. The BPW model also requires to construct a simulator, within the model, which we do not require. Therefore, we must be cautious with any comparison.

BPW-simulatability roughly states that $[\![P]\!] \approx P\|S$: the computational interpretation of the process P is indistinguishable from the simulated version of P. As shown in [7], this implies the trace mapping property, hence that symbolic trace properties transfer to the computational level. The BPW-simulatability should also imply the soundness of observational equivalence of the corresponding BPW-processes in a simple way (D. Unruh, private communication). The precise relationships with our work are worth being further investigated.

Conversely, if a simulated process $S\|P$ could be seen as the computational interpretation of a process Q, then the BPW-simulatability itself could be seen as an instance of our result.

Our work can also be seen as a generalization of soundness results for static equivalence [4, 3, 12] from a passive attacker to an active one. However, as we sketched above and as we will see on an example later, these results cannot be used directly in the active attacker case, which is the one we consider.

In [18] the authors show how to encode an equivalence property (actually computational secrecy for a given set of primitives) in the symbolic trace, using patterns. This allows to show how an indistinguishability property can be lifted to the symbolic case. The method, contrary to ours, is however dedicated to this particular property.

The work of Mitchell *et. al.* [33] also aims at faithfully abstracting the model of interactive Turing machines. Their results concern general processes and not only a small fragment, as we do here. In this respect, they are much more general than us. However, on the other hand, they abstract much less: there are still computations, coin tossing and probabilistic transitions in their model. Our aim is really to show that it makes no difference if the attacker is given only a fixed set of operations (encryption, decryption, name generation...) and if there is no probabilities nor coin tossing.

To our knowledge, the only previous work formally connecting observational equivalence and computational indistinguishability is [6]. In this paper, the authors give soundness and completeness results of a cryptographic implementation of processes. The major difference with our work is that they do not have explicit cryptographic constructions in the formal model. For instance encryption keys cannot be sent or leaked since they are implicit. Most standard security protocols cannot be described at this level of abstraction without possibly missing attacks. The results of [6] are useful in designing secure implementations of abstract functionalities, not for the verification of existing protocols.

Finally, the work on automation and decision of observational equivalence [25, 15, 24] shows that there exist systematic ways of deriving such equivalences in the symbolic world. This is also the advantage of using a standard process algebra as a symbolic model.

Organization of the paper: we first give the definitions of our computational model in section 2. Next we recall some of the general definitions of applied π-calculus in section 3. Note that, in the following, we only consider a fragment of the calculus for the protocol description (as usual), and we will only consider a particular equational theory corresponding to symmetric encryption. The relationship between the two models, as well as the protocol description language is given in section 4. In section 5 we give our main result and outline the proof. More details, including intermediate lemmas, the notions of computation trees, tree oracles, tree soundness are provided in section 6. We omit details and proofs in this short paper: they can be found in [20].

2. COMMUNICATING TURING MACHINES

Randomized Turing machines are Turing machines with an additional *random tape*. We assume w.l.o.g. that these machine first draw an infinite random input string on the random tape, and then compute deterministically. *Communicating Turing machines* are randomized machines equipped with input/output tapes and two special instructions: send and receive. They are restricted to work in polynomial time *with respect to their original input* (see [11] for a discussion). The adversary is a special CTM with an additional *scheduling tape*. A *network* $\mathcal{F}\|A$ consists of an adversary A and a family of CTMs $\mathcal{F} = (\mathcal{M}_n)_{n\in\mathbb{N}}$. We also call $\mathcal{F}$ the *environment* of A. This model is a simplification of interactive Turing machines of [17], keeping only the essential features.

In brief, in the *initial configuration*, each machine of the network has the security parameter in unary on its input tape and possibly other data such as secret keys. For simplicity we do not model here the key distribution. Moves between configurations are defined according to the attacker's view: in each configuration, the attacker decides to perform an internal move, to ask for the initialization of a new machine or to schedule a communication. In the latter case, the identity of the recipient is written on the scheduling tape and either a send or a receive action is performed. In case of a send, the contents of the sending tape is copied to the receiving tape of the scheduled recipient, who performs (in one single step) all its computations, until (s)he is waiting for another communication. In case of a receive action, the whole content of the sending tape of the scheduled machine is copied on the receiving tape of the attacker. The number of CTMs in the network is unbounded. Note that this setting does allow dynamically corrupted parties as in most results relating symbolic and computational models. Initially corrupted machines simply send their keys on the network.

We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *negligible* if, for every polynomial P, $\exists N \in \mathbb{N}, \forall \eta > N, f(\eta) < \frac{1}{P(\eta)}$. We write $\mathbf{Pr}\{x : P(x)\}$ the probability of event $P(x)$ when the sample x is drawn according to an appropriate distribution (the key distribution or the uniform distribution; this is kept implicit).

Two families of machines are indistinguishable if any adversary cannot tell with which family he is connected with non negligible probability:

DEFINITION 1. *Two environments $\mathcal{F}$ and $\mathcal{F}'$ are* indistinguishable*, which we write $\mathcal{F} \approx \mathcal{F}'$, if, for every attacker A,*

$$|\boldsymbol{Pr}\{\overline{r}, r : (\mathcal{F}(\overline{r})\|A(r))(0^\eta) = 1\} - \boldsymbol{Pr}\{\overline{r}, r : (\mathcal{F}'(\overline{r})\|A(r))(0^\eta) = 1\}|$$

is negligible. $\overline{r}$ is the sequence of random inputs of the machines in $\mathcal{F}$ (resp. $\mathcal{F}'$), including keys. r is the random input of the attacker.

As described in introduction, the *computational secrecy* of s can be expressed as follows. In $\mathcal{F}_0$, the machines using s are set with s_0 while in $\mathcal{F}_1$, they are set with s_1. The values s_0 and s_1 could also be chosen by the attacker. Then the data s is computationally secret if $\mathcal{F}_0 \approx \mathcal{F}_1$. Note that the environments $\mathcal{F}_b$ may contain corrupted machines, not holding s_i, that start by leaking their private information to the adversary.

Anonymity of group signatures [5] is defined through the following game: the adversary chooses a message m and two identities i_0 and i_1. Then in $\mathcal{F}_0$, the machines sign m with identity i_0 while in $\mathcal{F}_1$, the machines sign m with identity i_1. Again the property is defined by $\mathcal{F}_0 \approx \mathcal{F}_1$.

3. THE APPLIED PI-CALCULUS

We use the applied π-calculus of [2] as a symbolic model. There are several reasons for this choice. First, there are verification tools relying on this model [15]. Next, Though only a small fragment of this process calculus is used in

the present paper, we plan several extensions in various directions: considering more primitives (and equational theories), enriching the control structure, e.g. with conditionals and sequential composition,... The applied π-calculus is well suited for such extensions.

We recall the definitions in this section. Note that we will only consider a small fragment of the applied-π-calculus for the protocol descriptions and only a particular equational theory for our main result.

3.1 Syntax

A *signature* is a finite set of function symbols with an arity. It represents the security primitives (e.g. encryption, pairing, decryption). Given a signature Σ, an infinite set $\mathcal{N}$ of *names* and an infinite set $\mathcal{X}$ of *variables*, $\mathcal{T}(\mathcal{N},\mathcal{X})$ is the set of *terms*, the least set containing $\mathcal{N},\mathcal{X}$ and closed by application of a symbol in Σ. We assume that Σ contains a binary pairing function $< u, v >$, the corresponding projections functions π_1,π_2, and a length function l, which is a morphism from $T(\mathcal{N},\mathcal{X})$ to $\mathbb{N}$. We assume infinitely many names of any length. Terms represent messages and names stand for (randomly) generated data. $\alpha,\beta,\ldots$ are meta-variables that range over names and variables. We confuse the name generation and the local variables using the same ν construction, as they obey the same scoping/renaming rules. $\overline{u}$ stands for a sequence $u_1,\ldots,u_n$. $\sigma = \{x_1 \mapsto s_1,\ldots,x_k \mapsto s_k\}$ is the substitution that replaces the variable x_i with the term s_i. The domain of σ, denoted by $\mathsf{dom}(\sigma)$ is the set $\{x_1,\ldots,x_k\}$.

EXAMPLE 3.1. *Σ_0 is the signature consisting of the binary pairing $< \cdot,\cdot >$, the two associated projections π_1,π_2, the binary decryption* dec *and the ternary symbol $\{\cdot\}^{\cdot}_{\cdot}$ for symmetric encryption: $\{x\}^r_k$ stands for the encryption of x with the key k and the random r. Σ_0 also contains constants with in particular a constant 0^l of length l for every l.*

The syntax of processes and extended processes is displayed in Figure 1. In what follows, we restrict ourselves to processes with public channels. $\mathcal{P}$ is a set of predicate symbols with an arity. A difference with [2] is that we consider conditionals with arbitrary predicates. This leaves some flexibility in modeling various levels of assumptions on the cryptographic primitives. Typical examples are the ability (or not) to check whether a decryption succeeds, or the ability (or not) to check that two ciphertexts are produced using the same encryption key. Other examples are typing predicates, which we may want (or not). In [2] the condition is always an equality. Encoding the predicate semantics with equalities is (only) possible when there is no negative condition: it suffices then to express when a predicate is true. We believe that predicates allow to better reflect the ability of the adversary, with less coding. As we will see in the section 4, the predicates will be interpreted as polynomially computable Boolean functions.

Note that we use unbounded (un-guarded) replication of processes. This does not prevent from getting both soundness and completeness w.r.t. the computational interpretation: we show that if there is a computational attack, then there is a symbolic one (soundness). This symbolic attack does not depend on the security parameter: in this respect, it is a constant size attack. Interpreting back the attack in the computational world, this means that there is an attack whose size is independent of the security parameter.

ϕ,ψ ::=	conditions
$p(s_1,\ldots,s_n)$	predicate application
$\phi \wedge \psi$	conjunction
P,Q,R ::=	processes
$c(x).P$	input
$\overline{c}(s).P$	output
$\mathbf{0}$	terminated process
$P \parallel Q$	parallel composition
$!P$	replication
$(\nu\alpha)P$	restriction
`if` ϕ `then` P `else` Q	conditional
A,B,C ::=	extended processes
P	plain process
$A \parallel B$	parallel composition
$(\nu\alpha)A$	restriction
$\{x \mapsto s\}$	active substitution

Figure 1: Syntax of processes

In the paper, we often confuse "process" an "extended process" (and do not follow the lexicographic convention $A,B...$ vs $P.Q,...$).

3.2 Operational semantics

We briefly recall the operational semantics of the applied pi-calculus (see [2] for details). E is a set of equations on the signature Σ, defining an equivalence relation $=_E$ on $\mathcal{T}(\mathcal{N})$, which is closed under context. $=_E$ is meant to capture several representations of the same message. Predicate symbols are interpreted as relations over $\mathcal{T}(\mathcal{N})/=_E$. This yields a structure $\mathcal{M}$.

EXAMPLE 3.2. *The equations E_0 corresponding to Σ_0 are* $\mathsf{dec}(\{x\}^z_y, y) = x \qquad \pi_1(< x, y >) = x \qquad \pi_2(< x, y >) = y$

They can be oriented, yielding a convergent rewrite system: every term s has a unique normal form $s\downarrow$. We may also consider the following predicates:

- *M is unary and holds on a (ground) term s iff $s\downarrow$ does not contain any projection nor decryption symbols.*
- *EQ is binary and holds on s,t iff $M(s), M(t)$ and $s\downarrow = t\downarrow$: this is a strict interpretation of equality.*
- *$\mathsf{P}_{\mathsf{samekey}}$ is binary and holds on ciphertexts using the same encryption key: $\mathcal{M} \models \mathsf{P}_{\mathsf{samekey}}(s,t)$ iff $\exists k,u,v,r,r'.EQ(s,\{u\}^r_k) \wedge EQ(t,\{v\}^{r'}_k)$.*
- *EL is binary and holds on s,t iff $M(s), M(t)$ and s,t have the same length. we assume that there is a length function, which is defined on terms as a homomorphism from terms to natural numbers.*

The structural equivalence is the smallest equivalence relation on processes that is closed under context application and that satisfies the relations of Figure 2. $\mathsf{fn}(P)$ (resp. $\mathsf{fv}(P)$) is the set of names (resp. variables) that occur free in P. Bound names are renamed thus avoiding captures. $P\{x \mapsto s\}$ is the process P in which free occurrences of

$$
\begin{array}{rcll}
A \parallel \mathbf{0} & \equiv & A & \\
A \parallel B & \equiv & B \parallel A & \\
(A \parallel B) \parallel C & \equiv & A \parallel (B \parallel C) & \\
(\nu\alpha)(\nu\beta)A & \equiv & (\nu\beta)(\nu\alpha)A & \\
(\nu\alpha)(A \parallel B) & \equiv & A \parallel (\nu\alpha)B & \text{if } \alpha \notin \mathsf{fn}(A) \cup \mathsf{fv}(A) \\
(\nu x)\{x \mapsto s\} & \equiv & \mathbf{0} & \\
(\nu\alpha)\mathbf{0} & \equiv & \mathbf{0} & \\
!P & \equiv & P \parallel !P & \\
\{x \mapsto s\} \parallel A & \equiv & \{x \mapsto s\} \parallel A\{x \mapsto s\} & \\
\{x \mapsto s\} & \equiv & \{x \mapsto t\} & \text{if } s =_E t
\end{array}
$$

Figure 2: Structural equivalence

x are replaced by s. An *evaluation context* is a process $C = (\nu\overline{\alpha})([\cdot] \parallel P)$ where P is a process. We write $C[Q]$ for $(\nu\overline{\alpha})(Q \parallel P)$. A context (resp. a process) C is *closed* when $\mathsf{fv}(C) = \emptyset$ (there might be free names).

Possible evolutions of processes are captured by the relation $\rightarrow$, which is the smallest relation, compatible with the process algebra and such that:

$$
\begin{array}{llcll}
\text{(Com)} & c(x).P \parallel \overline{c}(s).Q & \rightarrow & \{x \mapsto s\} \parallel P \parallel Q & \\
\text{(Cond1)} & \mathtt{if}\ \phi\ \mathtt{then}\ P\ \mathtt{else}\ Q & \rightarrow & P & \text{if } \mathcal{M} \models \phi \\
\text{(Cond2)} & \mathtt{if}\ \phi\ \mathtt{then}\ P\ \mathtt{else}\ Q & \rightarrow & Q & \text{if } \mathcal{M} \not\models \phi
\end{array}
$$

$\xrightarrow{*}$ is the smallest transitive relation on processes containing $\equiv$ and $\rightarrow$ and closed by application of contexts. We write $P \xrightarrow{c(t)} Q$ (resp. $P \xrightarrow{\overline{c}(t)} Q$) if there exists P' such that $P \xrightarrow{*} c(x).P'$ and $\{x \mapsto t\} \| P' \xrightarrow{*} Q$ (resp. $P \xrightarrow{*} \overline{c}(t).P'$ and $P' \xrightarrow{*} Q$).

Definition 2. *The* observational equivalence relation $\sim_o$ *is the largest symmetric relation $\mathcal{S}$ on closed extended processes such that $A\mathcal{S}B$ implies:*

1. *if, for some context C, term s and process A', $A \xrightarrow{*} C[\overline{c}(s) \cdot A']$ then for some context C', term s' and process B', $B \xrightarrow{*} C'[\overline{c}(s') \cdot B']$.*
2. *if $A \xrightarrow{*} A'$ then, for some B', $B \xrightarrow{*} B'$ and $A'\mathcal{S}B'$*
3. *$C[A]\mathcal{S}C[B]$ for all closed evaluation contexts C*

Example 3.3 (Group signature). *Group signature as defined in [5] can be modeled as observational equivalence as follows. Let $P(x,i)$ be the protocol for signing message x with identity i. Let $P_1 = c(y).P(\pi_1(y), \pi_1(\pi_2(y)))$ and $P_2 = c(y).P(\pi_1(y), \pi_2(\pi_2(y)))$. Intuitively, the adversary will send $< m, < i_1, i_2 >>$ where m is a message to be signed and i_1, i_2 are two identities. P_1 signs m with i_1 while P_2 signs m with i_2. Then P preserves anonymity if $P_1 \sim_o P_2$.*

3.3 Simple processes

We do not need the full applied pi-calculus to symbolically represent CTMs. For example, CTMs do not communicate directly: all communications are controlled by the attacker and there is no private channel. Thus we consider the fragment of *simple processes* built on *basic processes*. Simple processes capture the usual fragment used for security protocols. A basic process represents a session of a protocol role where a party waits for a message of a certain form and when all equality tests succeed, outputs a message accordingly. Then the party waits for another message and so on. The sets of basic processes $\mathcal{B}(i, \overline{n}, \overline{x})$, where $\overline{x}$ is a variable sequence, i is a name, called *pid*, standing for the process id and $\overline{n}$ is a name sequence (including for instance fresh and long-term keys), are the least sets of processes such that $\mathbf{0} \in \mathcal{B}(i, \overline{n}, \overline{x})$ and

- If $B \in \mathcal{B}(i, \overline{n}, \overline{x})$, $s \in \mathcal{T}(\overline{n}, \overline{x})$, ϕ is a conjunction of EQ and M atomic formulas such that $\mathsf{fn}(\phi) \subseteq \overline{n}$ and $\mathsf{fv}(\phi) \subseteq \overline{x}$, $\perp$ is a special error message, then $\mathtt{if}\ \phi\ \mathtt{then}\ \overline{\mathsf{c_{out}}}(s) \cdot B\ \mathtt{else}\ \overline{\mathsf{c_{out}}}(\perp) \cdot \mathbf{0} \in \mathcal{B}(i, \overline{n}, \overline{x})$. Intuitively, if all tests are satisfied, the process sends a message depending on its inputs.
- if $B \in \mathcal{B}(i, \overline{n}, \overline{x}, x)$ and $x \notin \overline{x}$, then

  $\mathsf{c_{in}}(x).\ \mathtt{if}\ EQ(\pi_1(x), i)\ \mathtt{then}\ B\ \mathtt{else}\ \overline{\mathsf{c_{out}}}(\perp) \cdot \mathbf{0}$

  $\in \mathcal{B}(i, \overline{n}, \overline{x})$. Intuitively, on input x, the basic process first checks that it is the expected recipient of the message, before processing it.

$\mathsf{c_{out}}$ and $\mathsf{c_{in}}$ are two special names, representing resp. the send tape and the receive tape.

Example 3.4. *The Wide Mouth Frog [16] is a simple protocol where a server transmits a session key from an agent A to an agent B.*

$$
\begin{array}{rcl}
A \rightarrow S & : & A, \{N_a, B, K_{ab}\}_{K_{as}} \\
S \rightarrow B & : & \{N_b, A, K_{ab}\}_{K_{bs}}
\end{array}
$$

A session of role A played by agent a can be modeled by the basic process

$$
A(a,b) = \mathtt{if}\ \mathsf{true}\ \mathtt{then}\ \overline{\mathsf{c_{out}}}(< a, \{< n_a, < b, k_{ab} >>\}^r_{k_{as}} >) \cdot \mathbf{0}\ \mathtt{else}\ \overline{\mathsf{c_{out}}}(\perp) \cdot \mathbf{0}
$$

Similarly a session of role S played for agents a, b and whose id is l, can be modeled by

$$
\begin{array}{l}
S(a,b,l) = \mathsf{c_{in}}(x).\ \mathtt{if}\ EQ(\pi_1(x), l)\ \mathtt{then} \\
\mathtt{if}\ \pi_1(\pi_2(x)) = a\ \wedge\ \pi_1(\pi_2(\mathsf{dec}_{k_{as}}(\pi_2(\pi_2(x))))) = b\ \mathtt{then} \\
\overline{\mathsf{c_{out}}}(\{< n_b, < a, \pi_2(\pi_2(\mathsf{dec}_{k_{as}}(\pi_2(\pi_2(x))))) >>\}^r_{k_{bs}}) \cdot \mathbf{0} \\
\mathtt{else}\ \overline{\mathsf{c_{out}}}(\perp) \cdot \mathbf{0}\ \mathtt{else}\ \overline{\mathsf{c_{out}}}(\perp) \cdot \mathbf{0}
\end{array}
$$

A *simple process* combines any number of instances of the protocol roles, hiding names that are meant to be (possibly shared) secrets:

$$
\begin{array}{l}
(\nu\overline{n})[\ (\nu\overline{x_1}, \overline{n_1}B_1 \| \sigma_1) \| \cdots \| (\nu\overline{x_k}, \overline{n_k}B_k \| \sigma_k) \\
!(\nu\overline{y_1}, l_1, \overline{m_1\mathsf{c_{out}}}(l_1)B'_1)\ \|\ \cdots\ \|\ !(\nu\overline{y_n}, l_n, \overline{m_n}.\overline{\mathsf{c_{out}}}(l_n)B'_n)\ ]
\end{array}
$$

where $B_j \in \mathcal{B}(i_j, \overline{n} \uplus \overline{n_j}, \overline{x_j})$, $\mathsf{dom}(\sigma_j) \subseteq \overline{x_j}$, $B'_j \in \mathcal{B}(l_j, \overline{n} \uplus \overline{m_j}, \overline{y_j})$. Note that each basic process B'_j first publishes its identifier l_j, so that an attacker can communicate with it. Each process of the form $!((\nu\overline{y_i}, l_i)\overline{\mathsf{c_{out}}}(l_i).B'_i)$ is a *replicated process*.

In the definition of simple processes, we assume that for any subterm $\{t\}^v_k$ occurring in a simple process, v is a name that does not occur in any other term, and belongs to the set of restricted names $\overline{n}$. (Still, there may be several occurrences of $\{t\}^v_k$, unlike in [4]).

EXAMPLE 3.5. *Continuing Example 3.4, a simple process representing unbounded number of sessions in which A plays a (with b) and s plays S (with a, b) for the Wide Mouth Frog protocol is*

$$\nu(k_{as}, k_{bs}) \quad \Big(\quad !((\nu k_{ab}, n_a, r, l)\overline{\mathsf{Cout}}(l).A(a,b)) \\ \| \; !((\nu x, n_b, r, l)\overline{\mathsf{Cout}(l)}.S(a,b,l)) \quad \Big)$$

For simplicity, we have only considered sessions with the same agent names a, b.

3.4 Deduction and static equivalence

As in the applied pi calculus [2], message sequences are recorded in *frames* $\phi = \nu\overline{n}.\sigma$, where $\overline{n}$ is a finite set of names, and σ is a ground substitution. $\overline{n}$ stands for fresh names that are *a priori* unknown to the attacker.

Given a frame $\phi = \nu\overline{n}.\sigma$ that represents the information available to an attacker, a ground term s is *deducible*, which we write $\nu\overline{n}.(\sigma \vdash s)$ if $\sigma \vdash s$ can be inferred using the following rules:

$$\frac{}{\sigma \vdash s} \begin{array}{c}\text{if } \exists x \in \mathsf{dom}(\sigma) \\ \text{s.t. } x\sigma = s \\ \text{or } s \in \mathcal{N} \setminus \overline{n}\end{array} \qquad \frac{\sigma \vdash s_1 \quad \ldots \quad \phi \vdash s_\ell}{\sigma \vdash f(s_1, \ldots, s_\ell)} \; f \in \Sigma$$

$$\frac{\sigma \vdash s}{\sigma \vdash s'} \; \mathcal{M} \models s = s'$$

EXAMPLE 3.6. *Consider the signature and equational theory of example 3.2. Let* $\phi = \nu n_1, n_2, n_3, r_1, r_2, r_3.\sigma$ *with* $\sigma = \{x_1 \mapsto \{n_1\}^{r_1}_{k_1}, x_2 \mapsto < \{n_2\}^{r_2}_{n_1}, \{n_3\}^{r_3}_{n_2} >\}$. *Then* $\nu n_1, n_2, n_3, r_1, r_2, r_3.(\sigma \vdash n_3)$.

Deduction is not sufficient for expressing the attacker's knowledge. We have also to consider its distinguishing capabilities. Using the predicate symbols, we get the following slight extension of static equivalence:

DEFINITION 3 (STATIC EQUIVALENCE). *Let ϕ be a frame, p be a predicate and $s_1, \ldots, s_k$ be terms. We say that $\phi \models p(s_1, \ldots, s_k)$ if there exists $\overline{n}$ such that $\phi = \nu\overline{n}.\sigma$, $\mathsf{fn}(s_i) \cap \overline{n} = \emptyset$ for any $1 \le i \le k$ and $\mathcal{M} \models p(s_1, \ldots, s_k)\sigma$. We say that two frames $\phi_1 = \nu\overline{n}.\sigma_1$ and $\phi_2 = \nu\overline{n'}.\sigma_2$ are* statically equivalent*, and write $\phi_1 \sim \phi_2$ when $\mathsf{dom}(\phi_1) = \mathsf{dom}(\phi_2)$, and*

$$\forall s_1, \ldots, s_k \in T(\mathcal{N}, \mathcal{X}), \forall p \in \mathcal{P}. \\ \phi_1 \models p(s_1, \ldots, s_k) \Leftrightarrow \phi_2 \models p(s_1, \ldots, s_k).$$

EXAMPLE 3.7. *Consider (again for the theory of Example 3.2) the two frames* $\phi_1 = \nu n_1, r_1, r_2.\{x \mapsto \{\{k\}^{r_1}_{n_1}\}^{r_2}_{n_1}\}$ *and* $\phi_2 = \nu n_2, r_3.\{x \mapsto \{s\}^{r_3}_{n_2}\}$. *If s has the same length as* $\{k\}^{r_1}_{n_1}$, *then the two frames are statically equivalent*

4. COMPUTATIONAL INTERPRETATION

Given a security parameter η and a mapping τ from names to actual bitstrings of the appropriate length, which depends on η, the computational interpretation $[\![s]\!]^\tau_\eta$ of a term s is defined as a $\mathcal{F}$-homomorphism: for each function symbol f there is a polynomially computable function $[\![f]\!]$ and $[\![f(t_1, \ldots, t_n)]\!]^\tau_\eta \stackrel{\text{def}}{=} [\![f]\!]([\![t_1]\!]^\tau_\eta, \ldots, [\![t_n]\!]^\tau_\eta)$.

In addition, for names, $[\![n]\!]^\tau_\eta \stackrel{\text{def}}{=} \tau(n)$. Such an interpretation must be compatible with the equational theory: $\forall s, t, \eta, \tau.\ s =_E t \Rightarrow [\![s]\!]^\tau_\eta = [\![t]\!]^\tau_\eta$.

Similarly, each predicate symbol p gets a computational interpretation $[\![p]\!]$ as a PPT that outputs a Boolean value. This is extended to conditions, using the standard interpretation of logical connectives. Given an interpretation $\mathcal{M}$ of the predicates symbols in the symbolic model we assume that $[\![p]\!]$ is an *implementation* of this interpretation $p \subseteq (T(\mathcal{N}))^n$, that is

$\mathbf{Pr}\{(x_1, \ldots, x_n) \stackrel{R}{\leftarrow} [\![t_1, \ldots, t_n]\!]_\eta : [\![p]\!](x_1, \ldots, x_n) = 1 - b\}$

is negligible for any $t_1, \ldots, t_n$, where $b = 1$ if $\mathcal{M} \models p(t_1, \ldots, t_n)$ and 0 otherwise.

EXAMPLE 4.1. *Consider the predicate symbols of Example 3.2. Assume that the decryption and projection functions return an error message $\perp$ when they fail. Then here are possible interpretations of some predicates:*

- *$[\![M]\!]$ is the set of bitstrings, which are distinct from $\perp$. $[\![M]\!]$ implements M if the encryption scheme is* confusion-free *(a consequence of INT-CTXT [31]).*
- *$[\![EQ]\!]$ is the set of pairs of identical bitstrings, which are distinct from $\perp$. It is an implementation of EQ as soon as $[\![M]\!]$ implements M.*

Given a random tape τ and a security parameter η, a simple process P is implemented as expected. In particular, we assume that shared names are distributed to the expected machines in an initialization phase and random number are computed according to the random tape. The implementation of P is denoted by $[\![P]\!]^\tau_\eta$.

5. MAIN RESULT

5.1 Assumptions and result

Encryption scheme.

We assume that it is IND-CPA (more precisely "type 3"-secure of [4]) and INT-CTXT, as defined in [13]. Moreover, we assume that each time the adversary needs a new key, it requests it to the protocol (e.g. using a corrupted party). The parties are supposed to check that the keys they are using have been properly generated.

Key hierarchy.

A term u which occurs at least once in t at another position than a key or a random number (third argument in encryption) is called a *plaintext subterm* of t. E.g. k_1 and k_3 occur in plaintext in $< k_1, \{\{k_3\}^{r_2}_{k_2}\}^{r_1}_{k_1} >$ but not k_2. We say that k *encrypts* k' in a set of terms S if S contains a subterm $\{u\}^r_k$ such that k' is a plaintext subterm of u. We assume a *key hierarchy*, i.e. an ordering on private keys such that, for any execution of the protocol no key encrypts a greater key. If there is a key hierarchy, no key cycle can be created. Note that, when comparing two processes, the two key hierarchies do not need to be identical.

Parsing.

To ease parsing operations, we assume that the pairing, key generation and encryption functions add a typing tag (which can be changed by the attacker), which includes which key is used in case of encryption. This can be easily achieved by assuming that a symmetric key k consists of

two parts (k_1, k_2), k_1 being generated by some standard key generation algorithm and k_2 selected at random. Then one encrypts with k_1 and tags the ciphertext with k_2.

We are now ready to state our main theorem: observational equivalence implies indistinguishability.

THEOREM 4. *Let P_1 and P_2 be two simple processes such that each P_i admits a key hierarchy. Assume that the encryption scheme is joint IND-CPA and INT-CTXT. Then $P_1 \sim_o P_2$ implies that $[\![P_1]\!] \approx [\![P_2]\!]$.*

For example, anonymity of group signature as defined in section 2 is soundly abstracted by the property defined in Example 3.3. Computational secrecy as defined in section 2 can be soundly abstracted by strong secrecy: a secret x is *strongly secret* in P if $P(s) \sim_o P(s')$ for any term s, s'.

5.2 Overview of the proof

The rest of the paper is devoted to the proof sketch of Theorem 4.

A first approach.

Let us first show why the naive ideas do not work. Assume we have proved that any computational trace is an interpretation (for some sample input) of a symbolic trace. Assume moreover that we have a soundness result showing that, if $s_1, \ldots, s_n$ and $u_1, \ldots, u_n$ are two equivalent sequences of terms, then the distributions $[\![s_1, \ldots, s_n]\!]$ and $[\![u_1, \ldots, u_n]\!]$ are indistinguishable. Assume finally that the traces of P_1 and the traces of P_2 can be pairwise associated in statically equivalent traces (as a consequence of observational equivalence).

One could think that it is possible to conclude, pretending that $[\![P_2]\!] \approx [\![P_1]\!]$ since $[\![t_1]\!] \approx [\![t_2]\!]$ for each trace t_1 of P_1 and the corresponding trace t_2 of P_2. This is however incorrect. Indeed, an attacker can choose his requests (hence the trace) depending on the sample input. In the equivalence $[\![t_1]\!] \approx [\![t_2]\!]$, we use an average on the samples, while the choice of t_1 (and t_2), may depend on this sample: there is a circular dependency.

To be more concrete, here is a toy example. Assume that an attacker, given a random input τ, first gets $[\![s]\!]^\tau$ (in both experiments) and then, schedules his requests depending on the ith bit of $[\![s]\!]^\tau$: at the ith step, he will get t_i^j (resp. u_i^j in the second experiment), where j is the ith bit of $[\![s]\!]^\tau$. Assume that, for any sequence of bits $j_1, \ldots, j_n$,

$$[\![s, t_1^{j_1}, \ldots, t_n^{j_n}]\!] \approx [\![s, u_1^{j_1}, \ldots, u_n^{j_n}]\!]$$

but that, for the particular sample τ such that $[\![s]\!]^\tau = j_1 \cdots j_n$, the attacker outputs 1 on input $[\![s, t_1^{j_1}, \ldots, t_n^{j_n}]\!]^\tau$ and outputs 0 on input $[\![s, u_1^{j_1}, \ldots, u_n^{j_n}]\!]^\tau$. This may happen as the distributions could be indistinguishable while distinguished on one particular sample value. Note that, in the distribution equivalence, we draw again a sample, while the choice of $j_1, ..., j_n$ depended precisely of that sample. Then the attacker always outputs 1 in the first experiment since he precisely chose from τ the sequence $j_1, ...j_n$. Similarly, he always outputs 0 in the second experiment: he gets a significant advantage, distinguishing the two processes.

The example shows that we cannot simply use the soundness of static equivalence on traces. The idea is to consider trees labeled with terms, instead of sequences of terms. Then we do not commit first to a particular trace (as choosing $j_1, ..., j_n$ above). Considering such trees requires an extension of the results of Abadi and Rogaway, which are proved for sequences of terms.

Proof sketch.

We associate a tree T_P with each process P, which we call *process computation tree* and define symbolic and computational equivalences (denoted respectively $\sim$ and $\approx$) on process computation trees (see the definitions in the section 6). Such trees record the possible behaviors of the symbolic process, depending on the input they get from the environment: T_P is a labeled transition system, whose initial state is P. We use process computation trees as an intermediate step and show the following implications:

$$P \sim_o Q \Rightarrow T_P \sim T_Q \Rightarrow T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

$P \sim_o Q \Rightarrow T_P \sim T_Q$: (Lemma 7) It holds for any term algebra, relying however on the particular fragment of process algebra (simple processes). This is similar to the classical characterization of observational equivalence as labeled bisimilarity.

$T_P \sim T_Q \Rightarrow T_P \approx T_Q$: (Lemma 11) It uses the (tree) soundness in the ground case. This is a new concept, which generalizes the soundness of static equivalence from sequences to trees. It is necessary for the preservation of trace equivalences.

As a (very simple) example, consider the trees T_P and T_Q whose edges are labeled with any possible pair of symbolic messages. The path labeled $< u_1, v_1 >, \ldots, < u_n, v_n >$ yields a node labeled with $\{u_1\}_k^{r_1}, \ldots, \{u_n\}_k^{r_n}$ in T_P and yields a node labeled with $\{v_1\}_k^{r_1}, \ldots, \{v_n\}_k^{r_n}$. This corresponds to a Left-Right oracle of an IND-CPA game. The tree soundness states in this case that the two trees are indistinguishable: even if the attacker adaptatively chooses his requests (i.e. a path in the tree), he cannot make a difference between the two experiments. IND-CCA2 could be also expressed in this way. Here we consider more general experiments, specified by the two processes P, Q.

$T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$ (Lemma 13) It uses trace lifting: we need to prove that a computational trace is, with an overwhelming probability, an instance of a symbolic trace.

For instance in the above IND-CPA game, we cheated a little bit since the requests of the attacker were instances of symbolic requests, while in a true IND-CPA game they can be arbitrary bitstrings. This last step shows that it is actually not cheating: it does not make a significant difference (actually it does not make any difference at all in an IND-CPA game).

The two last implications are proved here in the context of pairing and symmetric encryption only. However, we believe that the use of computation trees and the way we get rid of encryption, can be extended to other primitives.

6. COMPUTATION TREES

We first define a general notion of trees that could serve to design oracles: the main purpose is to lift static equivalence (of frames) to trees, i.e. in an adaptative setting. Trees defined by the protocols (processes) are special cases, as we will see next. But we use the general definition in further transformations of the oracles.

6.1 General Computation Trees

Let $S = T(\mathcal{N})$ be a set of labels, typically a pair $< i, u >$ of a pid and a term for a request u to the process i, or a request to start a new process. For $i = 1, 2$, let $\phi_i = \nu\overline{n}\sigma_i$ be two frames. We write $t \in \phi_1$ if $t = x\sigma_1$ for some $x \in \mathsf{dom}(\sigma_1)$ and $\phi_1 \subseteq \phi_2$ if $x\sigma_1 = x\sigma_2$ for all $x \in \mathsf{dom}(\sigma_1)$.

DEFINITION 5. *A* computation tree T *is a mapping from a prefix closed subset of* S^* *(*$Pos(T)$*, the* positions *of* T*) to pairs* (P, ϕ) *where* P *is a simple process and* ϕ *is a ground frame over* $\mathcal{T}(\mathcal{N})$*. If* $p \in Pos(T)$ *and* $T(p) = (P, \phi)$*, we write* $\phi(T, p)$ *the frame* ϕ*. Moreover* T *must satisfy the following conditions:*

- *for every positions* p, q*, if* $p > q$*, then* $\phi(T, q) \subseteq \phi(T, p)$
- *for every position* $p \cdot t$*,* $t \in \phi(T, p \cdot t)$
- *for every positions* $p \cdot t, p \cdot u$*, if* $t =_E u$*, then* $t = u$*. This ensures that there is no two branches labeled with the same (equivalent) message.*

Such trees will be used in the next section to represent all possible behaviors of the processes, in a structured way. Since we have unbounded replication, the trees need not to be bounded in depth: there may be infinite paths. They may also be infinitely branching, as, at any time, the possible attacker's actions are unbounded

In this definition, positions need not to be closed lexicographically. The definition of static equivalence $\sim$ is extended to computation trees. $T|_p$ is the sub-tree of T at position p.

DEFINITION 6. $\sim$ *is the largest equivalence relation on computation trees such that if* $T_1 \sim T_2$*, then* $\phi(T_1, \epsilon) \sim \phi(T_2, \epsilon)$ *and there is an one-to-one mapping* β *from* $T(\mathcal{N})$ *to itself such that, for any length 1 position* t *of* T_1*,* $T_1|_t \sim T_2|_{\beta(t)}$*.*

Typically, requests sent by the adversary need not to be identical, but must be equivalent. Then β is a mapping, which depends on T_1, T_2, which associates the messages in the labels of T_1 with equivalent messages labeling T_2.

6.2 Process computation trees

We organize all possible symbolic executions of a simple process P in a tree T_P. Each node of T_P is labeled by (Q, ϕ) where Q is the current state of the process and ϕ represents the sequence of messages sent over the network by P so far.

Let $\mathcal{P} \equiv \nu\overline{n}, \nu\overline{x}.\ Q_1\|\sigma_1\| \cdots \|Q_N\|\sigma_N\|S$ be a simple process where $S = S_1\| \cdots \|S_k$ is the composition of a finite number of replicated processes S_i and every $Q_i \in \mathcal{B}(l_i, \overline{n_i}, \overline{x_i})$ is either $\mathbf{0}$ or a basic process $\mathsf{c_{in}}(x_i).P_i$ and σ_i is a ground substitution whose domain contains only free variables of P_i. Note that l_i is the pid of Q_i. The *process computation tree* $T_\mathcal{P}$ is defined as follows. The labeling and positions are defined by induction on the position length: $T_\mathcal{P}(\epsilon) = (\mathcal{P}, \mathtt{id})$ where $\mathtt{id}$ denotes the empty frame, and, for any $p \in \mathsf{Pos}(T_\mathcal{P})$, let

$$T_\mathcal{P}(p) = (\nu\overline{n}\nu\overline{x}.\ Q_1^p\|\sigma_1^p\| \cdots \|Q_N^p\|\sigma_N^p\|S\ ,\ \ \nu\overline{n}\sigma)$$

where each Q_j^p is either $\mathbf{0}$ or $\mathsf{c_{in}}(x_j^p).P_j^p$. Then $q = p \cdot \alpha \in \mathsf{Pos}(T_\mathcal{P})$ if $\alpha =< l_i, u >$, $Q_i^p \not\equiv \mathbf{0}$, $\nu\overline{n}\sigma \vdash \alpha$ and

$$Q_i^p \xrightarrow{\mathsf{c_{in}}(\alpha)} \xrightarrow{\overline{\mathsf{c_{out}}}(\alpha_1)\cdots\overline{\mathsf{c_{out}}}(\alpha_m)} Q_i^q\|\sigma_i^p\|\{x_i^p \mapsto \alpha\}$$

in which case

$$T_\mathcal{P}(p \cdot \alpha) = (\nu\overline{n}\nu\overline{x}.Q_1^q\|\sigma_1^q\| \cdots \|Q_N^q\|\sigma_N^q,\ \nu\overline{n}.\sigma \cup \{x_\alpha \mapsto \alpha\} \cup \bigcup_{i=1}^m \{x_{\alpha_i} \mapsto \alpha_i\})$$

where, for every $j \neq i$, $Q_j^q = Q_j^p$, $\sigma_j^q = \sigma_j^p$, Q_i^q is either $\mathbf{0}$ or $\mathsf{c_{in}}(x_i^q).P(x_i^q)$ and $\sigma_i^q = \sigma_i^p \circ \{x_i^p \mapsto \alpha\}$. This corresponds to the case where an attacker sends a message to an active process Q of pid l_i. The attacker may also ask for the initialization of a process. $S = S_1\| \cdots \|S_k$ and there is a fixed ordering on the S_j (which correspond to the roles of the protocol). Then $q = p \cdot \mathsf{new}_j \in \mathsf{Pos}(T_\mathcal{P})$, where new_j is a special constant $(1 \leq j \leq k)$. Let $S_j =!(\nu\overline{y}\nu l\ \overline{\mathsf{c_{out}}}(l).B)$. Then

$$T_\mathcal{P}(p \cdot \mathsf{new}_j) = (\nu\overline{n}\nu\overline{x}\nu\overline{y}.\ Q_1^p\|\sigma_1^p\| \cdots \|Q_N^p\|\sigma_N^p\|B\|S\ ,\ \nu\overline{n}\sigma \cup \{z \mapsto l\})$$

where z is a fresh variable and assuming by renaming that the names of $\overline{y}$ do not appear free in $Q_1^p\|\sigma_1^p\| \cdots \|Q_N^p\|\sigma_N^p$. Note that by construction, $B \in \mathcal{B}(l, \overline{n_l}, \overline{x_l})$ thus l is the identifier of B. l is first published such that the intruder can use it to schedule B.

A process computation tree is a computation tree. Observational equivalence yields equivalence between the corresponding process computation trees:

LEMMA 7. *Let* P *and* Q *be two simple processes. If* $P \sim_o Q$ *then* $T_P \sim T_Q$*.*

This does not follow directly from [2] (see the proof in [20]).

6.3 Scheduled computation trees

When all behaviors of a concrete attacker are instances of behaviors of a symbolic attacker, the concrete attacker can be seen as a machine which schedules the behavior of the symbolic attacker. That is what we try to capture here.

We assume that, given a security parameter η and a mapping τ from names to actual bitstrings, there is a *parsing function* κ_η^τ that maps bitstring to their symbolic representation. This parsing function is assumed to be total, using possibly constants or new names of the appropriate length.

For any symbolic computation tree T, and random tape τ, we let $\mathcal{O}_{T,\tau}$ be an oracle, whose replies depend on the tree T and the sample τ. When T is a process computation tree, the oracle can be simply understood as simulating the network and answering to attacker's messages. This is convenient for an intuitive understanding of $\mathcal{O}_{T,\tau}$, but we will transform the tree T later on. That is why we need a general definition.

Intuitively, the tree specifies how the oracle can be adaptively queried and what are its answers. This is formalized as follows.

When queried with m_n, after being queried successively with $m_1, \ldots, m_{n-1}$,

- the oracle first computes $\kappa_\eta^\tau(m_n)$. Let $r_j = \kappa_\eta^\tau(m_j)$ for $1 \leq j \leq n$.
- the oracle returns 0 if $r_1 \cdots r_n$ is not a position of T
- otherwise, let $\phi_1 = \nu\overline{n_1}\sigma_1 = \phi(T, r_1 \cdots r_{n-1})$, $\phi_2 = \nu\overline{n_2}\sigma_2 = \phi(T, r_1 \cdots r_n)$ be the labels of the two successive nodes of T. For any name k that occurs in σ_2 and not in σ_1, the oracle draws a random number $\tau(k)$ using its random tape τ. If $k \notin \overline{n_2}$ then the oracle returns $\tau(k)$ (the value is public in that case).

  Next, the oracle returns $[\![x\sigma_2]\!]^\tau$ for all $x \in \mathsf{dom}(\sigma_2) \setminus \mathsf{dom}(\sigma_1)$. Intuitively, the oracle replies by sending back

the (interpretation of the) terms labeling the target node of the tree that have not been already given.

In the last case, the oracle answer corresponds, in case T is a process computation tree, to the messages sent by the process answering the attacker's message.

DEFINITION 8. *Given two symbolic computation trees T_1, T_2, the two trees are* computationally indistinguishable*, which we write $T_1 \approx T_2$ if, for any PPT A, $|\boldsymbol{Pr}\{\tau : A^{\mathcal{O}_{T_1,\tau}}(0^\eta) = 1\} - \boldsymbol{Pr}\{\tau : A^{\mathcal{O}_{T_2,\tau}}(0^\eta) = 1\}|$ is negligible.*

7. MAIN STEPS OF THE PROOF

From now on, we fix the signature Σ_0, the equations E_0 and the predicate set $\mathcal{P}_0$, defined in Example 3.2 and in the following examples of Section 3.2.

7.1 Getting rid of encryption

The function Ψ_k on trees replaces terms under encryption by constants 0^l of the same length and is used later for stepwise simplifications:

$$\begin{array}{rcl} \Psi_k(n) & = & n \quad \text{if } n \text{ is a name or a constant} \\ \Psi_k(< t_1, t_2 >) & = & < \Psi_k(t_1), \Psi_k(t_2) > \\ \Psi_k(\{t\}^r_k) & = & \{0^{l(t)}\}^r_k \\ \Psi_k(\{t\}^r_{k'}) & = & \{\Psi_k(t)\}^r_{k'} \quad \text{if } k \neq k' \end{array}$$

Then Ψ_k is extended to computation trees by applying Ψ_k on the requests and frames. Intuitively, the underlying process remains the same but the adversary is given a view of the execution where any encryption by k has been replaced by an encryption of zeros by k. If k is not deducible, then an intruder is unable to make a difference:

LEMMA 9. *For any computation tree T and for any name k such that k is not deducible from any frame labeling a node of T, then $T \sim \Psi_k(T)$.*

The lemma is proved by choosing Ψ_k for the one-to-one function β.

Now, once every encryption has been replaced by encryption of zeros then static equivalence coincides with equality up-to name renaming:

LEMMA 10. *Let ϕ_1 and ϕ_2 be two frames such that for any subterm of ϕ_1 or ϕ_2 of the form $\{u\}^r_k$, we have $u = 0^l$ for some $l \in \mathbb{N}$. If $\mathsf{P}_{\mathsf{samekey}}$ is in the set of predicates, then $\phi_1 \sim \phi_2$ iff ϕ_1 and ϕ_2 are equal up-to name renaming.*

7.2 Soundness of static equivalence on trees

Static equivalence on process trees can be transferred at a computational level.

LEMMA 11. *Let P_1 and P_2 be two simple processes such that each P_i admits a key hierarchy. Let T_{P_i} be the process computation tree associated to P_i. If $T_{P_1} \sim T_{P_2}$ then $T_{P_1} \approx T_{P_2}$ or the encryption scheme is not joint IND-CPA and INT-CTXT.*

This key lemma is proved by applying the functions Ψ_k following the key ordering, to the trees T_{P_i}. We preserve equivalence on trees thanks to Lemma 9. If we find a key k such that $\Psi_k(T_{P_i}) \not\approx T_{P_i}$, then we can construct an attacker who breaks IND-CPA. Otherwise we are left to trees labeled with frames whose only subterms of the form $\{u\}^r_k$ are such that $u = 0^l$ for some l. In this case, we show that equivalence of such frames coincides with equality using Lemma 10.

7.3 Relating concrete and symbolic traces

We need here to show that concrete traces are, with overwhelming probability, interpretations of symbolic ones. We first define formally what it means.

Given $\mathcal{P}, \eta, \tau, A$, the behavior of the network $A\|[\![\mathcal{P}]\!]^\tau$ is deterministic.

Its computation can be represented as $m_1 \cdots m_n$, the sequence of messages sent by the adversary.

If T is a process computation tree and $p \in Pos(T)$, p *fully abstracts* the computation sequence $m_1 \cdots m_n$ if $p = u_1 \cdots u_n$ and, for every $j \leq n$, $[\![u_j]\!]^\tau = m_j$. In other words, p fully abstracts a computation sequence if it defines a symbolic trace whose interpretation is that computation sequence.

LEMMA 12. *Assume that the encryption scheme is INT-CTXT and IND-CPA. Let $\mathcal{P}$ be a simple process that admits a key ordering and $T_{\mathcal{P}}$ be its process computation tree. Let A be a concrete attacker. The probability over all samples τ, that any computational sequence of $A\|[\![\mathcal{P}]\!]^\tau$ is fully abstracted by some path p in $T_{\mathcal{P}}$, is overwhelming.*

To prove this lemma, we first simplify the trees by applying the functions Ψ_k thanks to Lemmas 11 and 9. Then, we investigate in which cases the adversary may produce traces that cannot be lifted and, in each situation, we break either INT-CTXT or IND-CPA.

In the last lemma, we simply apply the previous result. Since traces can be lifted, an attacker on the concrete processes is actually a scheduler of the computation trees, hence distinguishing the concrete processes amounts to distinguish the corresponding computation trees.

LEMMA 13. *Let P_1 and P_2 be two simple processes admitting a key hierarchy. Let T_{P_i} be the process computation tree associated to P_i. If the encryption scheme is joint IND-CPA and INT-CTXT, then $T_{P_1} \approx T_{P_2}$ implies that $[\![P_1]\!] \approx [\![P_2]\!]$.*

Theorem 4 is now a straightforward consequence of Lemma 7, 11 and 13.

8. EXTENSIONS

Following our proof scheme, we believe that our results can be extended to other security primitives, e.g. public-key encryption or signatures. We also wish to extend the simple process, for instance addind conditionals and sequential composition.

There are harder extensions. For instance, can we drop the requirement that private keys are not dynamically disclosed? As explained in [8], there is a commitment problem if we rely on a simulator. We could also extend our results to a wider class of equational theories by extending in particular Lemma 11.

Acknowledgements

We thank Michael Backes, Steve Kremer, Dominique Unruh and Bogdan Warinschi for their comments on this paper, as well as the anonymous referees.

9. REFERENCES

[1] M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static

equivalence. In *Foundations of Software Science and Computation Structure (FoSSaCS'06)*, volume 3921 of *LNCS*, pages 398–412, 2006.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.

[3] M. Abadi and J. Jürgens. Formal eavesdropping and its computational interpretation. In *Theoretical Aspects of Computer Software, LNCS 2215*, 2001.

[4] M. Abadi and P. Rogaway. Reconciling two views of cryptography: the computational soundness of formal encryption. *J. Cryptology*, 2007.

[5] M. Abdalla and B. Warinschi. On the minimal assumptions of group signature schemes. In *6th International Conference on Information and Communication Security*, pages 1–13, 2004.

[6] P. Adão and C. Fournet. Cryptographically sound implementations for communicating processes. In *International Colloquium on Algorithms, Languages and Programming (ICALP'06)*, 2006.

[7] M. Backes, M. Dürmuth, and R. Küsters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Proc. of 27th FSTTCS*, volume 4855 of *LNCS*, 2007.

[8] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Computer Security Foundations Workshop (CSFW'04)*, 2004.

[9] M. Backes and B. Pfitzmann. Relating cryptographic und symbolic key secrecy. In *Symp. on Security and Privacy (SSP'05)*, pages 171–182, 2005.

[10] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *10th ACM Concerence on Computer and Communications Security (CCS'03)*, 2003.

[11] M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, 2007.

[12] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. ICALP'05*, volume 3580 of *LNCS*, 2005.

[13] M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *LNCS*, pages 531–545, 2000.

[14] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Computer Security Foundations Workshop (CSFW'01)*, 2001.

[15] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[16] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report 39, Digital Systems Research Center, February 1989.

[17] R. Canetti. Universal composable security: a new paradigm for cryptographic protocols. In *Symposium on Foundations of Computer Science*, 2001.

[18] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols. In *Theory of Cryptography Conference (TCC'06)*, 2006.

[19] R. Canetti and T. Rabin. Universal composition with joint state. Cryptology ePrint Archive, report 2002/47, Nov. 2003.

[20] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. Research Report 6508, INRIA, `https://hal.inria.fr/inria-00274158`, Apr. 2008.

[21] V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In *Proc. FSTTCS*, volume 4337 of *LNCS*, pages 176–187, 2006.

[22] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 157–171, 2005.

[23] S. Delaune, S. Kremer, and M. D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, 2006.

[24] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi-calculus. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *LNCS*, pages 133–145, 2007.

[25] H. Hüttel. Deciding framed bisimilarity. In *Proc. INFINITY'02*, 2002.

[26] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 172–185. Springer, 2005.

[27] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In *advances in cryptology, (CRYPTO-97)*, volume 1294 of *LNCS*, pages 150–164, 1997.

[28] S. Kremer and L. Mazaré. Adaptive soundness of static equivalence. In *European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *LNCS*, pages 610–625, 2007.

[29] R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Computer Security Foundations (CSF'08)*, 2008.

[30] P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Symp. on Security and Privacy (SSP'04)*, pages 71–85, 2004.

[31] D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway language of encrypted expressions. *Journal of Computer Security*, 2004.

[32] D. Micciancio and B. Warinschi. Soundness of formal encryption in presence of an active attacker. In *Theory of Cryptography Conference (TCC'04)*, volume 2951 of *LNCS*, 2004.

[33] J. Mitchell, A. Ramanathan, and V. Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Comput. Sci.*, 353:118–164, 2006.

How to prove security of communication protocols? A discussion on the soundness of formal models w.r.t. computational ones.*

Hubert Comon-Lundh[1] and Véronique Cortier[2]

1 **ENS Cachan& CNRS & INRIA Saclay Ile-de-France**
2 **LORIA, CNRS**

Abstract

Security protocols are short programs that aim at securing communication over a public network. Their design is known to be error-prone with flaws found years later. That is why they deserve a careful security analysis, with rigorous proofs. Two main lines of research have been (independently) developed to analyse the security of protocols. On the one hand, formal methods provide with symbolic models and often automatic proofs. On the other hand, cryptographic models propose a tighter modeling but proofs are more difficult to write and to check. An approach developed during the last decade consists in bridging the two approaches, showing that symbolic models are *sound* w.r.t. symbolic ones, yielding strong security guarantees using automatic tools. These results have been developed for several cryptographic primitives (e.g. symmetric and asymmetric encryption, signatures, hash) and security properties.

While proving soundness of symbolic models is a very promising approach, several technical details are often not satisfactory. Focusing on symmetric encryption, we describe the difficulties and limitations of the available results.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Verification, security, cryptography

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.29

1 Introduction

Security protocols aim at securing communications over public networks. They are typically designed for bank transfers over the Internet, establishing private channels, or authenticating remote sites. They are also used in more recent applications such as e-voting procedures. Depending on the application, they are supposed to ensure security properties such as confidentiality, privacy or authentication, even when the network is (at least partially) controlled by malicious users, who may intercept, forge and send new messages. While the specification of such protocols is usually short and rather natural, designing a secure protocol is notoriously difficult and flaws may be found several years later. A famous example is the "man-in-the-middle" attack found by G. Lowe against the Needham-Schroder public key protocol [41]. A more recent example is the flaw discovered in Gmail (and now fixed) by Armando *et. al.* [9].

* This work has been supported by the ANR-07-SeSur-002 AVOTÉ project.

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 29–44

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

During the two last decades, formal methods have demonstrated their usefulness when designing and analyzing security protocols. They indeed provide with rigorous frameworks and techniques that allow to discover new flaws. For instance, the two previously mentioned flaws have been discovered while trying to prove the security of the protocol in a formal setting. Following the seminal work of Dolev and Yao [33], many techniques have been developed for analysing the security of protocols, often automatically. For example, the AVISPA platform [8] and the ProVerif tool [20] are both efficient and practical tools for automatically proving security properties or finding bugs if any. The security of protocols is undecidable in general [34]. Checking the secrecy and authentication-like properties is however NP-complete when the number of sessions is fixed [44]. Several extensions have been designed, considering more security properties or more security primitives [2, 25, 28, 24, 37]. Bruno Blanchet has developed an (incomplete) procedure based on clause resolution [19] for analyzing protocols for an unbounded number of sessions. All these approaches rely on a common representation for messages: they are symbolically modeled by *terms* where each function symbol represents a cryptographic primitive, some of their algebraic properties being reflected in an equational theory. Then protocols are modeled using or adapting existing frameworks such as fragments of logic, process algebras or constraint systems.

While the symbolic approaches were successful in finding attacks, the security proofs in these models are questionable, because of the level of abstraction: most cryptographic details are ignored. This might be a problem: for instance, it is shown in [45] that a protocol can be proved in a formal, symbolic, model, while there is an attack, that also exploits some finer details of the actual implementation of the encryption scheme. In contrast, cryptographic models are more accurate: the security of protocols is based on the security of the underlying primitives, which in turn is proved assuming the hardness of various computational tasks such as factoring or computing discrete logarithms. The messages are bitstrings. The proofs in the computational model imply strong guarantees (security holds in the presence of an arbitrary probabilistic polynomial-time adversary). However, security reductions for even moderately-sized protocols become extremely long, difficult, and tedious. Recently, a significant research effort [6, 43, 13, 15, 11, 26] has been directed towards bridging the gap between the symbolic and the cryptographic approaches. Such *soundness* results typically show that, under reasonable cryptographic assumptions such as IND-CCA2 for the encryption scheme, proofs in symbolic models directly imply proofs in the more detailed cryptographic models. These approaches are very promising: they allow to reconcile two distinct and independently developed views for modeling and analysing security protocols. Second and more importantly, they allow to obtain the best of the two worlds: strong security guarantees through the simpler symbolic models, that are amenable to automatic proofs.

However, such soundness results also assume many other properties regarding the implementation or even regarding the key infrastructure. In this paper, we discuss these usually under-looked assumptions, pointing the limitations of current results. In particular, we provide with several protocols (counter) examples, for which IND-CCA2 does not imply the security, as soon as a malicious user may chose its own keys at its will. These examples show that standard symbolic models are not sound w.r.t. cryptographic ones when using symmetric encryption. We also discuss how to symbolically represent the length of messages and what are the implications on the implementation. All these examples will be discussed within the the applied-pi calculus [3], but the counter-examples do not depend on this particular process algebra: the discussion will stay at a rather informal level and can be understood without familiarity with the applied-pi calculus.

Related work. Many soundness results have been established in various settings. We discuss some of them in Section 3. Fewer works are dedicated to the limitations. Backes and Pfitzmann have shown that primitives such as Exclusive Or or hash functions cannot be soundly abstracted in their simulatability library [14]. This is related to the impossibility of constructing some universally composable primitives [40]. This witnesses the difficulty of designing sound and accurate models for some primitives. [1] compares CryptoVerif [21], an automatic tool designed for performing proofs directly in the cryptographic model, and the use of soundness results, emphasizing the current limitations of the latter.

2 Setting

We recall here briefly part of the syntax and the operational semantics of the applied π-calculus of [3]. We are going to use a small fragment of this calculus for the formal definition of the protocols.

2.1 Syntax

In any symbolic model for security protocols, messages are modeled by *terms*, which are built on a set of function symbols Σ, that represent the cryptographic primitives (e.g. encryption, pairing, decryption). Given an infinite set $\mathcal{N}$ of *names* and an infinite set $\mathcal{X}$ of *variables*, $\mathcal{T}(\mathcal{N}, \mathcal{X})$ is the set of *terms*:

s, t, u ::=	terms	
x, y, z	variable	
a, b, c, k, n, r	name	
$f(s_1, \ldots, s_k)$	function application	$f \in \Sigma$ and k is the arity of f.

Terms represent messages and names stand for (randomly) generated data. We assume the existence of a length function l, which is a Σ-morphism from $T(\mathcal{N})$ to $\mathbb{N}$.

In what follows, we will consider symmetric encryption and pairing. Let Σ_0 consist of the binary pairing $< \cdot, \cdot >$, the two associated projections π_1, π_2, the binary decryption dec and the ternary symbol $\{\cdot\}_\cdot^\cdot$ for symmetric encryption: $\{x\}_k^r$ stands for the encryption of x with the key k and the random r. Σ_0 also contains constants, in particular a constant 0^l of length l for every l.

The syntax of processes is displayed in Figure 1. In what follows, we restrict ourselves to processes with public channels: there is no restriction on name channel. We assume a set $\mathcal{P}$ of predicate symbols with an arity. Such a definition, as well as its operational semantics coincides with [3], except for one minor point introduced in [26]: we consider conditionals with arbitrary predicates. This leaves some flexibility in modeling various levels of assumptions on the cryptographic primitives.

In what follows, we may use expressions of the form $\mathsf{let}\ \ldots\ \mathsf{in}\ \ldots$ as a syntactic sugar to help readability.

2.2 Operational semantics

We briefly recall the operational semantics of the applied pi-calculus (see [3, 26] for details). E is a set of equations on the signature Σ, defining an equivalence relation $=_E$ on $\mathcal{T}(\mathcal{N})$, which is closed under context. $=_E$ is meant to capture several representations of the same message. This yields a quotient algebra $\mathcal{T}(\mathcal{N})/=_E$, representing the messages. Predicate symbols are interpreted as relations over $\mathcal{T}(\mathcal{N})/=_E$. This yields a structure $\mathcal{M}$.

Φ_1, Φ_2 ::=	conditions
$p(s_1, \ldots, s_n)$	predicate application
$\Phi_1 \wedge \Phi_2$	conjunction
P, Q, R ::=	processes
$c(x).P$	input
$\overline{c}(s).P$	output
$\mathbf{0}$	terminated process
$P \parallel Q$	parallel composition
$!P$	replication
$(\nu\alpha)P$	restriction
`if` Φ `then` P `else` Q	conditional

Figure 1 Syntax of processes

In what follows, we will consider the equational theory E_0 on Σ_0 defined by the equations corresponding to encryption and pairing:

$$\mathsf{dec}(\{x\}^z_y, y) = x \qquad \pi_1(< x, y >) = x \qquad \pi_2(< x, y >) = y$$

These equations can be oriented, yielding a convergent rewrite system: every term s has a unique normal form $s\downarrow$.

We also consider the following predicates introduced in [26].

- M checks that a term is well formed. Formally, M is unary and holds on a (ground) term s iff $s\downarrow$ does not contain any projection nor decryption symbols and for any $\{u\}^r_v$ subterm of s, v and r must be names. This forbids compound keys for instance.
- EQ checks the equality of well-formed terms. EQ is binary and holds on s, t iff $M(s), M(t)$ and $s\downarrow = t\downarrow$: this is a strict interpretation of equality.
- $\mathsf{P}_{\mathsf{samekey}}$ is binary and holds on ciphertexts using the same encryption key: $\mathcal{M} \models \mathsf{P}_{\mathsf{samekey}}(s, t)$ iff $\exists k, u, v, r, r'.EQ(s, \{u\}^r_k) \wedge EQ(t, \{v\}^{r'}_k)$.
- EL is binary and holds on s, t iff $M(s), M(t)$ and s, t have the same length.

▶ **Example 2.1.** The Wide Mouth Frog [22] is a simple protocol where a server transmits a session key K_{ab} from an agent A to an agent B. This toy example is also used in [1] as a case study for both CryptoVerif and soundness techniques. For the sake of illustration, we propose here a flawed version of this protocol.

$$\begin{array}{lcl} A \to S & : & A, B, \{N_a, K_{ab}\}_{K_{as}} \\ S \to B & : & A, \{N_s, K_{ab}\}_{K_{bs}} \end{array}$$

The server is assumed to share long-term secret keys with each agent. For example, K_{as} denotes the long-term key between A and the server. In this protocol, the agent A establishes a freshly generated key K_{ab} with B, using the server for securely transmitting the key to B.

A session l_a of role A played by agent a with key k_{as} can be modeled by the process

$$A(a, b, k_{as}, l_a) \stackrel{\text{def}}{=} (\nu r, n_a)\, \overline{\mathsf{c_{out}}}(< l_a, < a, < b, \{< n_a, k_{ab} >\}^r_{k_{as}} >>>) \cdot \mathbf{0}$$

Similarly a session of role S played for agents a, b with corresponding keys k_{as} and k_{bs}, can be modeled by

$$S(a,b,k_{as},k_{bs},l_s) \stackrel{\text{def}}{=} (\nu n_s, r)\ \mathsf{c_{in}}(x).\ \texttt{if}\ EQ(\pi_1(x), l_s)\ \texttt{then}\ \mathsf{let}\ y = \pi_2(\mathsf{dec}(\pi_2(\pi_2(\pi_2(x))), k_{as}))\ \mathsf{in}$$
$$\texttt{if}\ \ \pi_1(\pi_2(x)) = a\ \wedge\ \pi_1(\pi_2(\pi_2(x)) = b\ \wedge\ M(y)\ \texttt{then}$$
$$\overline{\mathsf{c_{out}}}(< l_s, < a, \{< n_s, y >\}^r_{k_{bs}} >>) \cdot \mathbf{0}$$
$$\texttt{else}\ \overline{\mathsf{c_{out}}}(\bot)\cdot\mathbf{0}\ \texttt{else}\ \overline{\mathsf{c_{out}}}(\bot)\cdot\mathbf{0}$$

where l_s is the session identifier of the process.

Then an unbounded number of sessions of this protocol, in which A plays a (with b) and s plays S (with a, b and also with b, c) can be represented by the following process

$$P_{\mathsf{ex}} = \nu(k_{as}, k_{bs})\ \ (\ \ !((\nu k_{ab}, l_a)\overline{\mathsf{c_{out}}}(l_a).A(a,b,k_{as},l_a,r))$$
$$\|\ !((\nu l_s)\overline{\mathsf{c_{out}}}(l_s).S(a,b,k_{as},k_{bs},l_s))\ \|\ !((\nu l_s)\overline{\mathsf{c_{out}}}(l_s).S(a,c,k_{as},k_{cs},l_s))\ \ )$$

To reflect the fact that c is a dishonest identity, its long-term key k_{cs} shared with the server does not appear under a restriction and is therefore known to an attacker.

The environment is modeled through *evaluation context*, that is a process $C = (\nu\overline{\alpha})([\cdot]\|P)$ where P is a process. We write $C[Q]$ for $(\nu\overline{\alpha})(Q \parallel P)$. A context (resp. a process) C is *closed* when it has no free variables (there might be free names).

Possible evolutions of processes are captured by the relation $\rightarrow$, which is the smallest relation, compatible with the process algebra and such that:

(Com)	$c(x).P \parallel \overline{c}(s).Q$	$\rightarrow$	$\{x \mapsto s\} \parallel P \parallel Q$	
(Cond1)	$\texttt{if}\ \Phi\ \texttt{then}\ P\ \texttt{else}\ Q$	$\rightarrow$	P	if $\mathcal{M} \models \Phi$
(Cond2)	$\texttt{if}\ \Phi\ \texttt{then}\ P\ \texttt{else}\ Q$	$\rightarrow$	Q	if $\mathcal{M} \not\models \Phi$

$\stackrel{*}{\rightarrow}$ is the smallest transitive relation on processes containing $\rightarrow$ and some structural equivalence (e.g. reflecting the associativity and commutativity of the composition operator $\|$) and closed by application of contexts.

▶ **Example 2.2.** Continuing Example 2.1, we show an attack, that allows an attacker to learn k_{ab}, the key exchanged between a and b. Indeed, an attacker can listen to the first message $< l_a, < a, < b, \{< n_a, k_{ab} >\}^{r_1}_{k_{as}} >>>$ and replace it with $< l_a, < a, < c, \{< n_a, k_{ab} >\}^{r_1}_{k_{as}} >>>$. Thus the server would think that a wishes to transmit her key k_{ab} to c. Therefore it would reply with $< a, \{< n_s, k_{ab} >\}^{r_2}_{k_{cs}} >$. The attacker can then very easily decrypt the message and learn K_{ab}. This attack corresponds to the context

$$C_{\mathsf{attack}} \stackrel{\text{def}}{=} [\cdot] \parallel \mathsf{c_{out}}(x_{l_a}).\mathsf{c_{out}}(x_{l_s}).\mathsf{c_{out}}(x_{m_a}).\ \text{//listens to sessions ids and the first message}$$
$$\mathsf{let}\ y = \pi_2(\pi_2(x_{m_a}))\ \mathsf{in}$$
$$\overline{\mathsf{c_{in}}}(< x_{l_s}, < a, < c, y >>>).\ \text{//replays the message, with } b \text{ replaced by } c$$
$$\mathsf{c_{out}}(x_{m_s}).\ \text{//listens to the server's reply}$$
$$\mathsf{let}\ y' = \mathsf{dec}(\pi_2(\pi_2(x_{m_s})), k_{cs})\ \mathsf{in}\ \overline{\mathsf{c_{in}}}(\pi_2(y')).0\ \text{//outputs the secret}$$

Then the attack is reflected by the transitions $C_{\mathsf{attack}}[P_{\mathsf{ex}}] \stackrel{*}{\rightarrow} \overline{\mathsf{c_{out}}}(k_{ab}) \parallel Q$ for some process Q, yielding the publication of the confidential key k_{ab}.

2.3 Observational equivalence

Observational equivalence is useful to describe many properties such as confidentiality or authentication as exemplified in [5]. It is also crucial for specifying privacy related properties as needed in the context of electronic voting protocols [32].

▶ **Definition 2.3.** The *observational equivalence relation* $\sim_o$ is the largest symmetric relation $\mathcal{S}$ on closed extended processes such that $A\mathcal{S}B$ implies:

1. if, for some context C, term s and process A',
   $A \xrightarrow{*} C[\overline{c}(s) \cdot A']$ then for some context C', term s' and process B', $B \xrightarrow{*} C'[\overline{c}(s') \cdot B']$.
2. if $A \xrightarrow{*} A'$ then, for some B', $B \xrightarrow{*} B'$ and $A'\mathcal{S}B'$
3. $C[A]\mathcal{S}C[B]$ for all closed evaluation contexts C

▶ **Example 2.4** (Group signature)**.** The security of group signature has been defined in [7]. It intuitively ensures that an attacker should not be able to distinguish two signatures performed with two distinct identities when they belong to the same group. It can be modeled as observational equivalence as follows. Let $P(x, i)$ be the protocol for signing message x with identity i. Let $P_0 = c(y).P(\pi_1(y), \pi_1(\pi_2(y)))$ and $P_1 = c(y).P(\pi_1(y), \pi_2(\pi_2(y)))$. Intuitively, the adversary will send $< m, < i_0, i_1 >>$ where m is a message to be signed and i_0, i_1 are two identities. P_0 signs m with i_0 while P_1 signs m with i_1. Then P preserves anonymity iff $P_0 \sim_o P_1$.

2.4 Computational interpretation

We assume given an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ where $\mathcal{G}$ is the generating function for keys, $\mathcal{E}$ is the encryption function and $\mathcal{D}$ the decryption function. We also assume given a pairing function. The encryption, decryption, and pairing functions and their corresponding projectors form respectively the computational interpretation of the symbols $\{\cdot\}, \mathsf{dec}, <, >, \pi_1, \pi_2$. We assume that the decryption and projection functions return an error message $\perp$ when they fail. Then, given an interpretation τ of names as bitstrings, $[\![\cdot]\!]_\tau$ is the (unique) Σ-morphism extending τ to $T(\mathcal{N})$; $[\![t]\!]_\tau$ is the *computational interpretation* of t. When τ is randomly drawn, according to a distribution that depends on a security parameter η, we may write $[\![t]\!]_\eta$ for the corresponding distribution and $[\![t]\!]$ for the corresponding family of distributions. Then here are possible interpretations of the predicates:

- $[\![M]\!]$ is the set of bitstrings, which are distinct from $\perp$. Intuitively $[\![M]\!]$ implements M if the encryption scheme is *confusion-free* (a consequence of INT-CTXT [42]).
- $[\![EQ]\!]$ is the set of pairs of identical bitstrings, which are distinct from $\perp$. It is an implementation of EQ as soon as $[\![M]\!]$ implements M.
- $[\![\mathsf{P}_{\mathsf{samekey}}]\!]$ is the set of pairs of bitstrings that have the same encryption tag.
- $[\![EL]\!]$ is the set of pairs of bitstrings of same length.

Processes can also be interpreted as communicating Turing machines. Such machines have been introduced in [16, 38] for modeling communicating systems. They are probabilistic Turing machines with input/output tapes. Those tapes are intuitively used for reading and sending messages. We will not describe them here and we refer to [26] for more details.

Now, given a process P without replication, one can interpret it as a (polynomial time) communicating Turing machine. The computational interpretation of P is denoted by $[\![P]\!]$ and is intuitively defined by applying the computational counterpart of each function and

predicate symbols. Then the replicated process $!P$ can also be interpreted by letting the adversary play with as many copies of $[\![P]\!]$ as he wants.

Indistinguishability. In computational models, security properties are often stated as *indistinguishability* of games. Two families of machines are indistinguishable if an adversary cannot tell them apart except with non negligible probability.

A function $f : \mathbb{N} \to \mathbb{N}$ is *negligible* if, for every polynomial P, $\exists N \in \mathbb{N}, \forall \eta > N, f(\eta) < \frac{1}{P(\eta)}$. We write $\mathbf{Pr}\{x : P(x)\}$ the probability of event $P(x)$ when the sample x is drawn according to an appropriate distribution (the key distribution or the uniform distribution; this is kept implicit).

▶ **Definition 2.5.** Two environments $\mathcal{F}$ and $\mathcal{F}'$ are *indistinguishable*, denoted by $\mathcal{F} \approx \mathcal{F}'$, if, for every polynomial time communicating Turing Machine A (i.e. for any attacker),

$$|\mathbf{Pr}\{\overline{r}, r : (\mathcal{F}(\overline{r}) \parallel A(r))(0^\eta) = 1\} - \mathbf{Pr}\{\overline{r}, r : (\mathcal{F}'(\overline{r}) \parallel A(r))(0^\eta) = 1\}|$$

is negligible. $\overline{r}$ is the sequence of random inputs of the machines in $\mathcal{F}$ (resp. $\mathcal{F}'$), including keys. r is the random input of the attacker.

For example, anonymity of group signatures as discussed in Example 2.4 is defined in [7] through the following game: the adversary chooses a message m and two identities i_0 and i_1. Then in $\mathcal{F}_0$, the machines sign m with identity i_0 while in $\mathcal{F}_1$, the machines sign m with identity i_1. Then the anonymity is defined by $\mathcal{F}_0 \approx \mathcal{F}_1$. Note that, for $i = 1, 2$, $\mathcal{F}_i$ can be defined as $[\![P_i]\!]$, implementation of the process P_i of the Example 2.4.

More generally, security properties can be defined by specifying the ideal behavior P_{ideal} of a protocol P and requiring that the two protocols are indistinguishable. For example, in [4], authenticity is defined through the specification of a process where the party B magically received the message sent by the party A. This process should be indistinguishable from the initial one.

3 Soundness results

Computational models are much more detailed than symbolic ones. In particular, the adversary is very general as it can be any (polynomial) communicating Turing machine. Despite the important difference between symbolic and computational models, it is possible to show that symbolic models are *sound* w.r.t. computational ones.

3.1 A brief survey

There is a huge amount of work on simulatability/universal composability, especially the work of Backes *et. al.* and Canetti [23, 13, 15, 11]. When the ideal functionality is the symbolic version of the protocol, then the black-box simulatability implies the trace mapping property [11], therefore showing a safe abstraction. Such results can be applied to trace properties such as authentication but not to indistinguishability. In a recent paper [12], Backes and Unruh show that the whole applied-pi calculus can be embedded in CoSP, a framework in which they prove soundness of public-key encryption and digital signatures, again for trace properties.

Besides [26], which we discuss in more detail below, one of the only results that prove soundness for indistinguishability properties is [39], for some specific properties (see the end of section 4 for more details).

In a series of papers starting with Micciancio and Warinschi [43] and continued with e.g. [31, 36], the authors show trace mapping properties: for some selected primitives (public-key encryption and signatures in the above-cited papers) they show that a computational trace is an instance of a symbolic trace, with overwhelming probability. But, again, this does not show that indistinguishability properties can be soundly abstracted, except for the special case of computational secrecy that can be handled in [31] and also in [29] for hash functions in the random oracle model.

We refer to [30] for a more complete survey of soundness results.

3.2 Observational equivalence implies indistinguishability

The main result of [26] consists in establishing that observational equivalence implies indistinguishability:

▶ **Theorem 3.1.** *Let P_1 and P_2 be two simple processes such that each P_i admits a key hierarchy. Assume that the encryption scheme is joint IND-CPA and INT-CTXT. Then $P_1 \sim_o P_2$ implies that $[\![P_1]\!] \approx [\![P_2]\!]$.*

This result assumes some hypotheses, some of which are explicitly stated above and informally discussed below (the reader is referred to [26] for the full details). There are additional assumptions, that are discussed in more details in the next section.

Simple processes are a fragment (introduced in [26]) of the applied-pi calculus. It intuitively consists of parallel composition of (possibly replicated) *basic processes*, that do not involve replication, parallel composition or else branches. Simple processes capture most protocols without else branch, for an unbounded number of sessions. For example, the process P_{ex} introduced in Example 2.1 is a basic process.

The most annoying restriction is the absence of conditional branching. Ongoing works should overcome this limitation, at the price of some additional computational assumptions. But the extension to the full applied π-calculus is really challenging, because of possible restrictions on channel names. Such restrictions indeed allow "private" computations, of which an attacker only observes the computing time (which is not part of the model).

Key hierarchy ensures that no key cycles can be produced on honest keys, even with the interaction of the adversary. This hypothesis is needed because current security assumptions such as IND-CPA do not support key cycles (most encryption schemes are not provably secure in the presence of key cycles). In [26], it is assumed that there exists a strict ordering on key such that no key encrypts a greater key.

Checking such conditions, for any possible interaction with the attacker, is in general undecidable, though a proof can be found in many practical cases. (And it becomes decidable when there is no replication [27]).

No dynamic corruption assumes that keys are either immediately revealed (e.g. corrupted keys) or remain secret. Showing a soundness result in case of dynamic key corruption is a challenging open question, that might require stronger assumptions on the encryption scheme.

IND-CPA and INT-CTXT are standard security assumptions on encryption schemes. The IND-CPA assumption intuitively ensures that an attacker cannot distinguish the encryption of any message with an encryption of zeros of the same length. INT-CTXT ensures that an

adversary cannot produce a valid ciphertext without having the encryption key. IND-CPA and INT-CTXT are standard security assumptions [18].

4 Current limitations

We discuss in this section the additional assumptions of Theorem 3.1. Let us emphasize that such assumptions are not specific to this result: other soundness results have similar restrictions and/or provide with a weaker result.

4.1 Parsing

Parsing the bitstrings into terms is used in the proof of the soundness results; this function has actually to computable in polynomial time, since this is part the construction of a Turing machine used in a reduction. Therefore, Theorem 3.1 assumes that the pairing, key generation and encryption functions add a typing tag (which can be changed by the attacker), that indicates which operator has been used and further includes which key is used in case of encryption. This can be achieved by assuming that a symmetric key k consists of two parts (k_1, k_2), k_1 being generated by some standard key generation algorithm and k_2 selected at random. Then one encrypts with k_1 and tags the ciphertext with k_2.

These parsing assumptions are easy to implement and do not restrict the computational power of an adversary. Adding tags can only add more security to the protocol. However, current implementations of protocols do not follow these typing hypotheses, in particular regarding the encryption. Therefore Theorem 3.1 requires a reasonable but non standard and slightly heavy implementation in order to be applicable.

The parsing assumption might be not necessary. There are ongoing works trying to drop it.

4.2 Length function

As explained in Section 2, Theorem 3.1 assumes the existence of a length function l, which is a morphism from $T(\mathcal{N})$ to $\mathbb{N}$. This length function is needed to distinguish between ciphertexts of different lengths. For example, the two ciphertext $\{< n_1, n_2 >\}_k$ and $\{n_1\}_k$ should be distinguishable while $\{< n_1, n_2 >\}_k$ and $\{< n_1, n_1 >\}_k$ should not. There are however cases where it is unclear whether the ciphertexts should be distinguishable or not:

$$\nu k.\overline{\mathsf{c_{out}}}(\{< n_1, n_2 >\}_k) \overset{?}{\sim}_o \nu k.\overline{\mathsf{c_{out}}}(\{\{n_1\}_k\}_k).$$

Whether these two ciphertexts are distinguishable typically depends on the implementation and the security parameter: their implementation may (or not) yield bitstrings of equal length. However, for a soundness result, we need to distinguish (or not) these ciphertexts *independently* of the implementation.
In other words, if we let length be the length of a bitstring, we (roughly) need the equivalence:

$$l(s) = l(t) \qquad \text{iff} \qquad \forall \tau.\ \mathsf{length}([\![s]\!]_\tau) = \mathsf{length}([\![t]\!]_\tau)$$

This requires some length-regularity of the cryptographic primitives. But even more, this requires length to be homogenous w.r.t. the security parameter η. To see this, consider the case where length is an affine morphism:

$$\begin{aligned} \mathsf{length}([\![\{t_1\}_k^r]\!]_\tau) &= \mathsf{length}([\![t_1]\!]_\tau) + \gamma \times \eta + \alpha \\ \mathsf{length}([\![< t_1, t_2 >]\!]_\tau) &= \mathsf{length}([\![t_1]\!]_\tau) + \mathsf{length}([\![t_2]\!]_\tau) + \beta \\ \mathsf{length}(\tau(n)) &= \delta \times \eta \end{aligned}$$

where β cannot be null since some bits are needed to mark the separators between the two strings $[\![t_1]\!]$ and $[\![t_2]\!]$. (Also, $\gamma, \delta > 0$.)

Now, if we consider an arbitrary term t, $\mathsf{length}([\![t]\!]_\tau) = n_1 \times \gamma \times \eta + n_1 \times \alpha + n_2 \times \beta + n_3 \times \delta \times \eta$. $\frac{\mathsf{length}([\![s]\!]_\tau)}{\mathsf{length}([\![t]\!]_\tau)}$ must be independent of η, hence there must exist $\alpha', \beta' \in \mathbb{N}, \beta' > 0$ such that $\alpha = \alpha' \times \eta$ and $\beta = \beta' \times \eta$.

This implies in particular that the pairing function always adds η bits (or a greater multiple of η) to the bitstrings. Similarly, a ciphertext should be the size of its plaintext plus a number of bits which is the a multiple of η.

While it is possible to design an implementation that achieves such constraints, this is not always the case in practice and it may yield a heavy implementation, in particular in conjunction with the parsing assumptions. Moreover, on the symbolic side, adding a length function raises non trivial decidability issues.

Adding a symbolic length function is needed for proving indistinguishability as illustrated by the former examples. It is worth noticing that several soundness results such as [13, 15, 31, 29, 12] do not need to consider a length function. The reason is that they focus on *trace properties* such as authentication but they cannot considered indistinguishability-based properties (except computational secrecy for some of them).

4.3 Dishonest keys

Theorem 3.1 assumes the adversary only uses correctly generated keys. In particular, the adversary cannot choose his keys at its will, depending on the observed messages. The parties are supposed to check that the keys they are using have been properly generated. The assumption could be achieved by assuming that keys are provided by a trusted server that properly generates keys together with a certificate. Then when a party receives a key, it would check that it comes with a valid certificate, guaranteeing that the key has been issued by the server. Of course, the adversary could obtain from the server as many valid keys as he wants.

However, this assumption is strong compared to usual implementation of symmetric keys and it is probably the less realistic assumptions among those needed for Theorem 3.1. We discuss alternative assumptions at the end of this section. It is worth noticing that in all soundness results for asymmetric encryption, it is also assumed that the adversary only uses correctly generated keys. Such an assumption is more realistic in an asymmetric setting as a server could certify public keys. However, this does not reflect most current implementations for public key infrastructure, where agents generate their keys on their own.

We now explain why such an assumption is needed to obtain soundness.The intuitive reason is that IND-CCA does not provide any guarantee on the encryption scheme when keys are dishonestly generated.

▶ **Example 4.1.** Consider the following protocol. A sends out a message of the form $\{c\}_{K_{ab}}$ where c is a constant. This can be formally represented by the process

$$A = (\nu r)\overline{\mathsf{c_{out}}}(< c, \{c\}^r_{k_{ab}} >).\mathbf{0}$$

Then B expects a key y and a message of the form $\{\{b\}_y\}_{K_{ab}}$ where b is the identity of B, in which case, it sends out a secret s (or goes in a bad state).

$$\begin{aligned} A &\to B: \quad (\nu r)\, c, \{c\}^r_{K_{ab}} \\ B: k, \{\{b\}_k\}_{K_{ab}} &\to A: \quad s \end{aligned}$$

This can be formally modeled by the process

$$B = \mathsf{c_{in}}(z).\ \texttt{if}\ \ EQ(b, \mathsf{dec}(\mathsf{dec}(\pi_2(z), k_{ab}), \pi_1(z)))\ \texttt{then}\ \overline{\mathsf{c_{out}}}(s)\ \texttt{else}\ \mathbf{0}$$

Then symbolically, the process $(\nu k_{ab})(\nu s)A \| B$ never emits s. However, a computational adversary can forge a key k such that any bitstring can be successfully decrypted to b using k. In particular, at the computational level, we have $\mathsf{dec}(c, k) = b$. Thus by sending $< k, \{c\}^r_{k_{ab}} >$ to B, the adversary would obtain the secret s.

This is due to the fact that security of encryption schemes only provides guarantees on *properly generated* keys. More precisely, given an IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, it is easy to build another IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$, which allows to mount the previous attack: consider $\mathcal{G}' = 0 \cdot \mathcal{G}$ (all honest keys begin with the bit 0), $\mathcal{E}'(m, i.k) = \mathcal{E}(m, k)$ for $i \in \{0, 1\}$ and $\mathcal{D}'$ defined as follows:

- $\mathcal{D}'(c, k) = \mathcal{D}(c, k')$ if $k = 0 \cdot k'$
- $\mathcal{D}'(c, k) = k'$ if $k = 1 \cdot k'$

It is easy to check that $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ remains IND-CCA2 and allows an adversary to choose a key that decrypts to anything he wants.

To capture this kind of computational attacks, an idea (from M. Backes [10]) is to enrich the symbolic setting with a rule that allows an intruder, given a ciphertext c and a message m, to forge a key such that c decrypts to m. This could be modeled e.g. by adding a functional symbol $\mathsf{fakekey}$ of arity 2 together with the equation

$$\mathsf{dec}(x, \mathsf{fakekey}(x, y)) = y$$

Going back to Example 4.1, this would allow a symbolic intruder to send the message $< \mathsf{fakekey}(c, b), \{c\}^r_{k_{ab}} >$ to the B process and the process $(\nu k_{ab})(\nu s)A \| B$ would emit s.

This solution appears however to be insufficient to cover the next examples.

▶ **Example 4.2** (hidden cyphertext)**.** The same kind of attacks can be mounted even when the ciphertext is unknown to the adversary. We consider a protocol where A sends $<< A, k >, \{\{k'\}^{r'}_k\}^r_{k_{ab}} >$ where k and k' are freshly generated keys. B recovers k' and sends it encrypted with k_{ab}. In case A receives her name encrypted with k_{ab}, A emits a secret s.

$$\begin{array}{rll} A & \to B: & (\nu k, k', r_1, r_2)\ A, k, \{\{k'\}^{r_1}_k\}^{r_2}_{K_{ab}} \\ B & \to A: & (\nu r_3)\ \{k'\}^{r_3}_{K_{ab}} \\ A : \{A\}_{K_{ab}} & \to B: & s \end{array}$$

Then symbolically, the process $(\nu k_{ab})(\nu s)A \| B$ would never emit s while again, a computational adversary can forge a key k such that any bitstring can be successfully decrypted to a using k.

This attack could be captured, allowing the forged key to be independent of the ciphertext. This can be modeled by the equation

$$\mathsf{dec}(x, \mathsf{fakekey}(y)) = y$$

where $\mathsf{fakekey}$ is now a primitive of arity 1. Some attacks may however require the decryption to depend from the cyphertext as shown in the next example.

▶ **Example 4.3** (simultaneous cyphertexts)**.** Consider the following protocol where A sends to B p cyphertexts $c_1, \ldots, c_p$. Then B encrypts all ciphertexts with a shared key k_{ab} together

with a fresh value n_b and commits to p other nonces $N_1, \ldots, N_p$. Then A simply forwards the cyphertext together with a fresh key k. Then B checks whether each cyphertext c_i decrypts to N_i using the key k received from A, in which case he sends out a secret s.

$$\begin{aligned} A &\to B: \quad c_1, \ldots, c_p \\ B &\to A: \quad \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\ A &\to B: \quad \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, k \\ B: \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}, k &\to A: s \end{aligned}$$

Then symbolically, the process $(\nu k_{ab})(\nu s)A \| B$ would never emit s since the N_i are generated after having received the c_i and the nonce N_b protects the protocol from replay attacks. However, having seen the c_i and the N_i, a computational adversary can forge a key k such that each bitstring c_i can be successfully decrypted to N_i using k. More precisely, given an IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, it is easy to build another IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$, which allows to mount the previous attack. Indeed, consider $\mathcal{G}' = 0 \cdot \mathcal{G}$ (all honest keys begin with the bit 0), $\mathcal{E}'(m, i.k) = \mathcal{E}(m, k)$ for $i \in \{0, 1\}$ and $\mathcal{D}'$ defined as follows:

- $\mathcal{D}'(c, k) = \mathcal{D}(c, k')$ if $k = 0 \cdot k'$
- $\mathcal{D}'(c, k) = n$ if $k = 1 \cdot c_1, n_1, \cdots c, n \cdots c_p, n_p$
- $\mathcal{D}'(c, k) = \perp$ otherwise

For dishonest keys, the decryption function $\mathcal{D}'(c, k)$ searches for c in k and outputs the following component when c is found in k. It is easy to check that $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ remains IND-CCA2 and allows an adversary to chose a key that decrypts to anything he wants, but with different possible outputs depending on the ciphertext.

To capture this attack, we need to consider a symbol of arity $2p$ for any p and an equation of the form

$$\mathsf{dec}(x_i, \mathsf{fakekey}(x_1, \ldots, x_p, y_1, \ldots, y_p)) = y_i$$

But this is still not be sufficient as the outcome may also depend on the ciphertext that is under decryption and on public data. Intuitively, decrypting with an adversarial key may produce a function depending on the underlying plaintext and on any previously known data.

Related work. To our best understanding of [13], these examples seem to form counter-examples of the soundness results for symmetric encryption as presented in [13]. An implicit assumption that solves this issue [10] consists in forbidding dishonest keys to be used for encryption or decryption (the simulator would stop as soon as it received a dishonest keys). As a consequence, only protocols using keys as nonces could be proved secure.

The only work overcoming this limitation in a realistic way is the work of Kuesters and Tuengerthal [39] where the authors show computational soundness for key exchange protocols with symmetric encryption, without restricting key generation for the adversary. Instead, they assume that sessions identifiers are added to plaintexts before encryption. This assumption is non standard but achievable. It would however not be sufficient in general as shown by the examples. In their case, such an assumption suffices because the result is tailored to key exchange protocols and realization of a certain key exchange functionality.

5 Conclusion

Among all the limitations we discuss in this paper, the main one is to consider only honestly generated keys (or a certifying infrastructure), which is completely unrealistic. There are

(at least) two main ways to overcome this assumption. A first possibility, already sketched in the paper, consists in enriching the symbolic model by letting the adversary create new symbolic equalities when building new (dishonest) keys. In this way, many protocols should still be provably secure under the IND-CCA assumption, yet benefiting from a symbolic setting for writing the proof.

A second option is to seek for stronger security assumptions by further requesting non-malleability. The idea is that a ciphertext should not be opened to a different plaintext, even when using dishonest keys. This could be achieved by adding a commitment to the encryption scheme [35].

However all these limitations also demonstrate that it is difficult to make symbolic and computational models coincide. Even for standard security primitives, soundness results are very strong since they provide with a generic security proof for *any* possible protocol (contrary to CryptoVerif). For primitives with many algebraic properties like Exclusive Or or modular exponentiation, the gap between symbolic and computation models is even larger and would require a lot of efforts.

We still believe that computational proofs could benefit from the simplicity of symbolic models, yielding automated proofs. An alternative approach to soundness results could consist in computing, out of a given protocol, the minimal computational hypotheses needed for its security. This is for example the approach explored in [17], though the symbolic model is still very complex.

Acknowledgement

We wish to thank Michael Backes and Dominique Unruh for very helpful explanations on their work and David Galindo for fruitful discussion on non-malleable encryption schemes.

References

1 M. Abadi, B. Blanchet, and H. Comon-Lundh. Models and proofs of protocol security: A progress report. In A. Bouajjani and O. Maler, editors, *Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 35–49, Grenoble, France, June-July 2009. Springer.

2 M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, November 2006.

3 M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

4 M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148(1), 1999.

5 M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

6 M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS2000)*, pages 3–22, August 2000.

7 M. Abdalla and B. Warinschi. On the minimal assumptions of group signature schemes. In *6th International Conference on Information and Communication Security*, pages 1–13, 2004.

8 A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb,

M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.

9 A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra. Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In *6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10, Alexandria, VA, USA, 2008.

10 M. Backes. Private communication, 2007.

11 M. Backes, M. Dürmuth, and R. Küsters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2007.

12 M. Backes, D. Hofheinz, and D. Unruh. CoSP a general framework for computational soundness proofs. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, pages 66–78, 2009.

13 M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Science Foundations Workshop (CSFW'04)*, pages 204–218, 2004.

14 M. Backes and B. Pfitzmann. Limits of the Cryptographic Realization of Dolev-Yao-style XOR and Dolev-Yao-Style Hash Functions. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, Lecture Notes in Computer Science, pages 336–354, 2005.

15 M. Backes and B. Pfitzmann. Relating cryptographic und symbolic key secrecy. In *26th IEEE Symposium on Security and Privacy*, pages 171–182, Oakland, CA, 2005.

16 M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, 2007.

17 G. Bana, K. Hasebe, and M. Okada. Secrecy-oriented first-order logical analysis of cryptographic protocols. Cryptology ePrint Archive, Report 2010/080, 2010. `http://eprint.iacr.org/`.

18 M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, 2000.

19 B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.

20 B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*, July 2005.

21 B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, Oct.–Dec. 2008. Special issue IEEE Symposium on Security and Privacy 2006. Electronic version available at http://doi.ieeecomputersociety.org/10.1109/TDSC.2007.1005.

22 M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society*, volume 426 of *Series A*, pages 233–271. 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

23 R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO 2001*, pages 19–40, Santa Barbara, California, 2001.

24 Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. *Theoretical Computer Science*, 338(1-3):247–274, June 2005.

25 Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Deciding the security of protocols with Diffie-Hellman exponentiation and product in exponents. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, 2003.

26 H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. ACM Press, 2008.

27 H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 11(4):496–520, 2010.

28 H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 271–280, Ottawa, Canada, June 2003. IEEE Computer Society Press.

29 V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In N. Garg and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187, Kolkata, India, December 2006. Springer.

30 V. Cortier, S. Kremer, and B. Warinschi. A Survey of Symbolic Methods in Computational Analysis of Cryptographic Systems. *Journal of Automated Reasoning (JAR)*, 2010.

31 V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, U.K, April 2005. Springer.

32 S. Delaune, S. Kremer, and M. D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, 2006.

33 D. Dolev and A. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of ComputerScience*, pages 350–357. IEEE Computer Society Press, 1981.

34 N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.

35 D. Galindo, F. D. Garcia, and P. Van Rossum. Computational soundness of non-malleable commitments. In *Proceedings of the 4th international conference on Information security practice and experience*, ISPEC'08, pages 361–376. Springer-Verlag, 2008.

36 R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185. Springer, 2005.

37 R. Küsters and T. Truderung. On the automatic analysis of recursive security protocols with xor. In *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS'07)*, volume 4393 of *LNCS*, Aachen, Germany, 2007. Springer.

38 R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Computer Security Foundations (CSF'08)*, 2008.

39 R. Küsters and M. Tuengerthal. Computational Soundness for Key Exchange Protocols with Symmetric Encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, pages 91–100. ACM Press, 2009.

40 Y. Lindell. General composition and universal composition in secure multiparty computation. In *Proc. 44th IEEE Symp. Foundations of Computer Science (FOCS)*, 2003.

41 G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, march 1996.

42 D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway language of encrypted expressions. *Journal of Computer Security*, 2004.

43 D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, February 2004. Springer-Verlag.

44 M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete. *Theoretical Computer Science*, 299:451–475, April 2003.

45 B. Warinschi. A computational analysis of the needham-schroeder protocol. In *16th Computer security foundation workshop (CSFW)*, pages 248–262. IEEE, 2003.

J Autom Reasoning (2011) 46:261–291
DOI 10.1007/s10817-010-9186-x

Automated Proofs for Asymmetric Encryption

J. Courant · M. Daubignard · C. Ene · P. Lafourcade · Y. Lakhnech

Received: 16 June 2010 / Accepted: 17 June 2010 / Published online: 3 July 2010

Abstract Many generic constructions for building secure cryptosystems from primitives with lower level of security have been proposed. Providing security proofs has also become standard practice. There is, however, a lack of automated verification procedures that analyze such cryptosystems and provide security proofs. In this paper, we present a sound and automated procedure that allows us to verify that a generic asymmetric encryption scheme is secure against chosen-plaintext attacks in the random oracle model. It has been applied to several examples of encryption schemes among which the construction of Bellare–Rogaway 1993, of Pointcheval at PKC'2000.

Keywords Provable cryptography · Asymmetric encryption · Automated verification · Hoare logic

1 Introduction

Our day-to-day lives increasingly depend upon information and our ability to manipulate it securely. This requires solutions based on cryptographic systems (primitives and protocols). In 1976, Diffie and Hellman invented public-key cryptography [15], coined the notion of one-way functions and discussed the relationship between cryptography and complexity theory. Shortly after, the first cryptosystem with a reductionist security proof appeared [20]. The next breakthrough towards formal proofs of security was the adoption 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 polynomial-time reduction proof from a

This work is partially supported by the ANR projects SCALP, AVOTE and SFINCS.

J. Courant · M. Daubignard · C. Ene · P. Lafourcade (✉) · Y. Lakhnech
Université Joseph Fourier (Grenoble 1), CNRS, Verimag, Grenoble, France
e-mail: pascal.lafourcade@imag.fr

hard problem to an attack against the security of the system. 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. While research in the field of provable cryptography has achieved tremendous progress towards rigorously defining the functionalities and requirements of many cryptosystems, little has been done for developing computer-aided proof methods or more generally for investigating a proof theory for cryptosystems as it exists for imperative programs, concurrent systems, reactive systems, etc.

In this paper, we present an automated proof method for analyzing generic asymmetric encryption schemes in the random oracle model (ROM). Generic encryption schemes aim at transforming schemes with weak security properties, such as one-wayness, into schemes with stronger security properties, specifically security against chosen ciphertext attacks. Examples of generic encryption schemes are [7, 8, 13, 18, 19, 21, 23, 25]. In this paper we propose a compositional Hoare logic for proving IND-CPA security. An important feature of our method is that it is not based on a global reasoning as it is the case for the game-based approach [9, 22]. Instead, it is based on local reasoning. Indeed, both approaches can be considered complementary as the Hoare logic-based one can be considered as aiming at characterizing by means of predicates the set of contexts in which the game transformations can be applied safely. In future work [11], we also present a method for proving plaintext awareness (PA). Plaintext awareness together with IND-CPA security imply IND-CCA security [3]. Combining the results of this paper with plaintext awareness, leads to a proof method for verifying the constructions in [7, 18, 19].

Related work We restrict our discussion to work aiming at providing computational proofs for cryptosystems. In particular, this excludes symbolic verification. We mentioned above the game-based approach [9, 17, 22]. B. Blanchet and D. Pointcheval developed a dedicated tool, CryptoVerif, that supports security proofs within the game-based approach [5, 6]. From the theoretical point of view, the main differences in our approaches are the following. CryptoVerif is based on observational equivalence. The equivalence relation induces rewriting rules applicable in contexts that satisfy some properties. Invariants provable in our Hoare logic can be considered as logical representations of these contexts. Moreover, as we are working with invariants, that is we follow a state-based approach, we need to prove results that link our invariants to game-based properties such as indistinguishability (cf. Propositions 1 and 3). G. Barthe, J. Cederquist and S. Tarento were among the first to provide machine-checked proofs of cryptographic schemes without relying on the perfect cryptography hypothesis. They have provided formal models of the Generic Model and the Random Oracle Model in the Coq proof assistant, and used this formalization to prove hardness of the discrete logarithm [1], security of signed ElGamal encryption against interactive attacks [10], and of Schnorr signatures against forgery attacks [24]. Recently in [4], the authors have been developing CertiCrypt, which provides support for formalizing game-based proofs in the Coq proof assistant. They have used their formalization to give machine-checked proofs of IND-CPA for OAEP. Another interesting piece of work to mention is the Hoare-style proof system proposed by R. Corin and J. Den Hartog for game-based cryptographic proofs [12]. Yet, there is no computer-assistance for the developed logic. In [14], Datta et al. present a computationally sound compositional logic (PCL) for key exchange

protocols. PCL has been applied successfully to the IEEE 802.11i wireless security standard and the IETF GDOI standard. PCL is a computationally sound Hoare-like logic for indistinguishability; one important difference is that, being focused on protocols, PCL does not provide support for standard cryptographic constructions such as one-way functions.

Outline In Section 2, we introduce notions used in the rest of the paper. In Section 3 we define our programming language and generic asymmetric encryption schemes. In Section 4, we present our Hoare logic for proving IND-CPA security. In Section 5, we explain how we can automate our procedure. Finally we conclude in Section 6 .

2 Preliminaries

In this section, we recall some basic definitions as well as poly-time indistinguishability. To do so, let us first introduce the following notations. If d is a distribution over a set E, we use $v \xleftarrow{r} d$ to denote that an element $v \in E$ is sampled according to distribution d. Moreover, as usual, if $d_1, \ldots, d_n$ are distributions, $[u_1 \xleftarrow{r} d_1; \ldots; u_n \xleftarrow{r} d_n : (u_{i_1}, \ldots, u_{i_k})]$ denotes the distribution obtained by performing the samplings in the order they are written and returning the result which is specified after the semi-colon.

We are interested in analyzing generic schemes for asymmetric encryption assuming ideal hash functions. That is, we are working in the *random oracle model* [7, 16]. Using standard notations, we write $H \xleftarrow{r} \Omega$ to denote that H is chosen uniformly at random from the set of functions with appropriate domain. By abuse of notation, for a list $\mathbf{H} = H_1, \cdots, H_n$ of hash functions, we write $\mathbf{H} \xleftarrow{r} \Omega$ instead of the sequence $H_1 \xleftarrow{r} \Omega, \ldots, H_n \xleftarrow{r} \Omega$. We fix a finite set $\{H_1, \ldots, H_n\}$ of hash functions and also a finite set Π of trapdoor permutations. We assume an arbitrary but fixed ordering on Π and $\mathbf{H}$; this allows us to freely switch between set-based and vector-based notation. A *distribution ensemble* is a countable sequence of distributions $\{X_\eta\}_{\eta \in \mathbb{N}}$ over states, $\mathbf{H}$ and Π. We only consider distribution ensembles that can be constructed in polynomial time in η by probabilistic algorithms that have oracle access to $\mathbf{H}$—this set is formally defined in the next section. Given two distribution ensembles $X = \{X_\eta\}_{\eta \in \mathbb{N}}$ and $X' = \{X'_\eta\}_{\eta \in \mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A}, \eta, X, X') = \left| \mathsf{Pr}\left[x \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(x) = 1\right] - \mathsf{Pr}\left[x \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(x) = 1\right] \right|$$

The probabilities are taken over X and the random coins used by the probabilistic adversary $\mathcal{A}$. We insist, above, that for each hash function H, the probabilities are indeed taken over the set of maps with the appropriate type. Two distribution ensembles X and X' are called *indistinguishable*, denoted by $X \sim X'$, if $\mathsf{Adv}(\mathcal{A}, \eta, X, X')$ is negligible as a function of η, for any polynomial-time (in η) probabilistic algorithm $\mathcal{A}$. All security notions we are going to use are in the ROM, where all algorithms, including adversaries, are equipped with oracle access to the hash functions.

3 A Simple Programming Language for Encryption and Decryption Oracles

In this section, we fix a notation for specifying encryption schemes. The notation we choose is a simple imperative language with random assignment. The language does not include loops since loops are usually not used for generic encryption schemes. Moreover, whereas the language allows the application of a trapdoor permutation f, it does not include the application of the inverse of a permutation. This choice is due to the fact that our analysis is carried out on encryption algorithms only, which, in the case of generic encryption schemes, seldom use permutation inverses.

Let Var be a finite set of variables. We assume that Var is large enough to deal with the considered examples. It should be obvious that our results do not depend on the size of Var. Our programming language is built according to the following BNF described in Table 1, where:

- $x \xleftarrow{r} \mathcal{U}$ samples a value in $\mathcal{U}_\eta$ and assigns it to x. Here, $(\mathcal{U}_\eta)_\eta$ is the family of uniform distributions over the set of bit-strings of length $\tau(x, \eta)$, where $\tau(x, \eta)$ is a polynomial in η.
- $x := f(y)$ applies the trapdoor one-way function f to the value of y and assigns the result to x.
- $x := H(y)$ applies the hash function H to the value of y and assigns the result to x. As a side effect, the pair $(v, H(v))$, where v is the value of y, is added to the variable $\mathbb{T}_H$ which stores the queries to the hash function H.
- $x := y \oplus z$ applies the exclusive or operator to the values of y and z and assigns the result to x.
- $x := y||z$ represents the concatenation of the values of y and z.
- $c_1; c_2$ is the sequential composition of c_1 and c_2.
- $\mathcal{N}(x, y)$: **var** $x_1; \cdots ; x_n;$ c is a procedure declaration, where $\mathcal{N}$ is its name (identifier for encryption or decryption procedure), **var** $x_1; \cdots ; x_n;$ declares the local variables $x_1, \cdots , x_n$, c is the body of the procedure and x and y are the input and output variables, respectively.

Example 1 The following command encodes the encryption scheme proposed by Bellare and Rogaway in [7] (shortly $f(r)||\text{in}_e \oplus G(r)||H(\text{in}_e||r)$):

$$
\begin{array}{l}
\mathcal{E}(\text{in}_e, \text{out}_e) : \\
\textbf{var } r; a; g; b; s; c; \\
r \xleftarrow{r} \{0,1\}^\eta;\ a := f(r);\ g := G(r); \\
b := \text{in}_e \oplus g;\ s := \text{in}_e||r;\ c := H(s); \\
\text{out}_e := (a||b)||c; \quad (\text{where } f \in \Pi \text{ and } G, H \in \mathbf{H})
\end{array}
$$

Semantics In addition to the variables in Var, we consider variables $\mathbb{T}_{H_1}, \ldots, \mathbb{T}_{H_n}$. Variable $\mathbb{T}_{H_i}$ is used to record the queries to the hash function H_i. Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to

Table 1 Language grammar

Command	$\mathsf{c} ::= x \xleftarrow{r} \mathcal{U} \mid x := f(y) \mid x := H(y) \mid x := y \oplus z \mid x := y\|\|z \mid c; c$
Oracle declaration	$\mathcal{O} ::= \mathcal{N}(x, y) :$ **var** $x_1; \cdots ; x_n;$ c

$\mathbb{T}_{H_i}$. Let $\mathbb{T}_{\mathbf{H}} = \{\mathbb{T}_{H_1}, \ldots, \mathbb{T}_{H_n}\}$. The reader should notice that the variables in $\mathbb{T}_{\mathbf{H}}$ are not in Var, and hence, cannot occur in commands.

We assume that all variables range over bit-strings (or pairs of bit-strings) with polynomial size in the security parameter η, so that domains of the variables in Var have a cardinality exponential in η. Given a state S, $S(\mathbb{T}_H).\mathsf{dom}$, respectively $S(\mathbb{T}_H).\mathsf{res}$, denotes the list obtained by projecting each pair in $S(\mathbb{T}_H)$ to its first, respectively second, element. Also we extend $S(\cdot)$ to expressions in the usual way, for instance $S(y \oplus z) = S(y) \oplus S(z)$, etc.

A program takes as input a *configuration* $(S, \mathbf{H}, (f, f^{-1}))$ and yields a distribution on configurations. A configuration is composed of a state S, a vector of hash functions $(H_1, \ldots, H_n)$ and a pair (f, f^{-1}) of a trapdoor permutation and its inverse, drawn thanks to a generator denoted $\mathbb{F}$. Let Γ denote the set of configurations and $\textsc{Dist}(\Gamma)$ the set of distributions on configurations. The semantics is given in Table 2, where $\delta(x)$ denotes the Dirac measure, i.e. $\mathsf{Pr}[x] = 1$ and $S(\mathbb{T}_H) \cdot (S(y), v)$ denotes the concatenation to $\mathbb{T}_H$ of a query to the hash function and its answer. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma)$ to $\textsc{Dist}(\Gamma)$. That is, let $F : \Gamma \to \textsc{Dist}(\Gamma)$ be a function. Then, F defines a unique function $F^* : \textsc{Dist}(\Gamma) \to \textsc{Dist}(\Gamma)$ such that $F^*(D) = [\gamma \xleftarrow{r} D; \gamma' \xleftarrow{r} F(\gamma) : \gamma']$. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

It is easy to prove that commands preserve the values of $\mathbf{H}$ and (f, f^{-1}). Therefore, we can, without ambiguity, write $S' \xleftarrow{r} [\![c]\!](S, \mathbf{H}, (f, f^{-1}))$ instead of $(S', \mathbf{H}, (f, f^{-1})) \xleftarrow{r} [\![c]\!](S, \mathbf{H}, (f, f^{-1}))$. According to our semantics, commands denote functions that transform distributions on configurations to distributions on configurations. However, only distributions that are constructible are of interest.

A family X of distributions is called *constructible*, if there is an algorithm A such that

$$X_\eta = \left[(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \mathbf{H} \xleftarrow{r} \Omega; S \xleftarrow{r} A^{\mathbf{H}}(f, 1^\eta) : (S, \mathbf{H}, f, f^{-1})\right],$$

where $\mathbb{F}$ is a *trapdoor permutation generator* that on input η generates an η-bit-string trapdoor permutation pair (f, f^{-1}), and A is a poly-time probabilistic algorithm with oracle access to the hash function, and such that A's queries to the hash oracles are recorded in the lists $\mathbb{T}_H$'s in S. The latter condition should not be understood as a restriction on the set of considered adversaries. Indeed, it does not mean that the adversaries need to honestly record their queries in the $\mathbb{T}_H$ lists. It rather means that in our reduction proofs, when we simulate such adversaries, we need to record their

Table 2 The semantics of the programming language

$$[\![x \xleftarrow{r} \mathcal{U}]\!](S, \mathbf{H}, (f, f^{-1})) = [u \xleftarrow{r} \mathcal{U} : (S\{x \mapsto u\}, \mathbf{H}, (f, f^{-1}))]$$

$$[\![x := f(y)]\!](S, \mathbf{H}, (f, f^{-1})) = \delta(S\{x \mapsto f(S(y))\}, \mathbf{H}, (f, f^{-1}))$$

$$[\![x := H(y)]\!](S, \mathbf{H}, (f, f^{-1})) =$$
$$\begin{cases} \delta(S\{x \mapsto v\}, \mathbf{H}, (f, f^{-1})) \text{ if } (S(y), v) \in \mathbb{T}_H \\ \delta(S\{x \mapsto v, \mathbb{T}_H \mapsto S(\mathbb{T}_H) \cdot (S(y), v)\}, \mathbf{H}, (f, f^{-1})) \text{ if } (S(y), v) \notin \mathbb{T}_H \text{ and } v = \mathbf{H}(H)(S(y)) \end{cases}$$

$$[\![x := y \oplus z]\!](S, \mathbf{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y) \oplus S(z)\}, \mathbf{H}, (f, f^{-1}))$$

$$[\![x := y||z]\!](S, \mathbf{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y)||S(z)\}, \mathbf{H}, (f, f^{-1}))$$

$$[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]$$

$$[\![\mathcal{N}(x, y) : \textbf{var } \mathbf{x}; c]\!](S, \mathbf{H}, (f, f^{-1})) = ([\![c]\!](S, \mathbf{H}, (f, f^{-1})))\{\mathbf{x} \mapsto S(\mathbf{x})\}$$

queries. We emphasize that the algorithm denoted above by A can, but does not necessarily represent (only) an adversary, e.g. it can be the sequential composition of an adversary and a command. We denote by $\textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ the set of constructible families of distributions. Distributions that are element of a constructible family of distributions are called *constructible* too.

Notice that $[\![c]\!](X) \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, for any command c and $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$.

3.1 Generic Asymmetric Encryption Schemes

We are interested in generic constructions that convert any trapdoor permutation into a public-key encryption scheme. More specifically, our aim is to provide an automatic verification method for generic encryption schemes.

Definition 1 A pair $(\mathbb{F}, \mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var } \mathbf{x}; \mathsf{c})$ defines a *generic encryption scheme*, where:

- $\mathbb{F}$ is a *trapdoor permutation generator* that on input η generates an η-bit-string trapdoor permutation pair (f, f^{-1}),
- $\mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var } \mathbf{x}; \mathsf{c}$ is a procedure declaration that describes the *encryption algorithm*. The variables in $\mathbf{x}$ are the local variables of the encryption algorithm. We require that the outcome of $\mathcal{E}(\text{in}_e, \text{out}_e)$ only depends on the value of in_e. Formally, for a triple $(S, \mathbf{H}, (f, f^{-1}))$, let $\textsc{Out}(S, \mathbf{H}, (f, f^{-1}))$ denote the distribution:

$$\left[(S', \mathbf{H}, (f, f^{-1})) \xleftarrow{r} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S, \mathbf{H}, (f, f^{-1})) : S'(\text{out}_e)\right].$$

  Then, we require that $\textsc{Out}(S, \mathbf{H}, (f, f^{-1})) = \textsc{Out}(S', \mathbf{H}, (f, f^{-1}))$ holds, for every states S and S' such that $S(\text{in}_e) = S'(\text{in}_e)$.

Let S_0 be the state that associates the bit-string $0^{\tau(x,\eta)}$, for any variable $x \in \mathsf{Var}$ that ranges over $\{0,1\}^{\tau(x,\eta)}$, and the empty list $[\,]$ with each variable in $\mathbb{T}_H$. Then, the usual definition of the IND-CPA security criterion (e.g. see [3]) can be stated as follows:

Definition 2 Let $GE = (\mathbb{F}, \mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var } \mathbf{x}; \mathsf{c})$ be a generic encryption scheme and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For $\eta \in \mathbb{N}$, let

$$\begin{aligned}\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta) = |2.\mathsf{Pr}[&(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \mathbf{H} \xleftarrow{r} \Omega;\\ &(m_0, m_1, \sigma) \xleftarrow{r} \mathcal{A}_1^{\mathbf{H}}(f);\\ &b \xleftarrow{r} \{0,1\};\\ &S' \xleftarrow{r} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S_0\{\text{in}_e \mapsto m_b\}, \mathbf{H}, (f, f^{-1})) :\\ &\mathcal{A}_2^{\mathbf{H}}(f, m_0, m_1, \sigma, S'(\text{out}_e)) = b\,] - 1|\end{aligned}$$

We insist, above, that $\mathcal{A}_1$ outputs bit-strings m_0, m_1 such that $|m_0| = |m_1|$, and an internal state σ to be forwarded to its guess-phase $\mathcal{A}_2$.

We say that GE is IND-CPA secure if $\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Notice that because of the condition put on generic encryption schemes, the choice of the state S_0 is not crucial. In other words, we can replace S_0 by any other state, we then get the same distribution of $S'(\text{out}_e)$ as a result. In fact, an equivalent formulation of Definition 2, consists in considering that the 1st-phase adversary $\mathcal{A}_1$ outputs a state S, such that messages m_0 and m_1 are stored in variables x_0 and x_1 and that σ is stored in variable x_σ:

$$\begin{aligned}\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta) = |2.\mathsf{Pr}[&(f,\ f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta);\ \mathbf{H} \xleftarrow{r} \Omega;\\ &S \xleftarrow{r} \mathcal{A}_1^{\mathbf{H}}(f);\\ &b \xleftarrow{r} \{0,1\};\\ &S' \xleftarrow{r} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S\{\text{in}_e \mapsto s(x_b)\}, \mathbf{H}, (f,\ f^{-1})) :\\ &\mathcal{A}_2^{\mathbf{H}}(f, S(x_0), S(x_1), S(x_\sigma), S'(\text{out}_e)) = b\,] - 1|\end{aligned}$$

Thus, we arrive at the following more appropriate equivalent definition of IND-CPA:

Definition 3 Let $GE = (\mathbb{F}, \mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var x}; \mathsf{c})$ be a generic encryption scheme and $\mathcal{A}$ be an adversary and $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$. For $\eta \in \mathbb{N}$, let

$$\begin{aligned}\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta, X_\eta) = 2.\mathsf{Pr}[&(S, \mathbf{H}, (f,\ f^{-1})) \xleftarrow{r} X_\eta;\ b \xleftarrow{r} \{0,1\};\\ &S' \xleftarrow{r} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S\{\text{in}_e \mapsto S(x_b)\}, \mathbf{H}, (f,\ f^{-1})) :\\ &\mathcal{A}^{\mathbf{H}}(f, S(x_0), S(x_1), S(s_\sigma), S'(\text{out}_e)) = b\,] - 1\end{aligned}$$

We say that GE is IND-CPA secure, if $\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta, X_\eta)$ is negligible for any constructible distribution ensemble X and polynomial-time adversary $\mathcal{A}$.

4 A Hoare Logic for IND-CPA Security

In this section, we present our Hoare logic for proving IND-CPA security. We prove that the presented logic is sound. In addition to axioms that deal with each basic command and operation, random assignment, concatenation, xor, etc..., our logic includes the usual sequential composition and consequence rules of the Hoare logic. In order to apply the consequence rule, we use entailment (logical implication) between assertions as in Lemma 2.

Our Hoare logic can be easily transformed into a procedure that allows us to prove properties by computing invariants of the encryption oracle. More precisely, the procedure annotates each control point of the encryption command with a set of predicates that hold at that point for any execution. Given an encryption oracle $\mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var x}; \mathsf{c}$ we want to prove that at the final control point, we have an invariant that tells us that the value of out_e is indistinguishable from a random value. Classically, this implies IND-CPA security.

First, we present the assertion language we use to express the invariant properties we are interested in. Then, we present a set of rules of the form $\{\varphi\}\mathsf{c}\{\varphi'\}$, meaning that execution of command c in any distribution that satisfies φ leads to a distribution that satisfies φ'. Using Hoare logic terminology, this means that the triple $\{\varphi\}\mathsf{c}\{\varphi'\}$ is valid. From now on, we suppose that the adversary has access to the hash functions $\mathbf{H}$, and he is given the trapdoor permutation f, but not its inverse f^{-1}.

4.1 The Assertion Language

Before specifying the assertion language, we give a few definitions and notations that we use to define the predicates of the language.

Definition 4 The set of variables used as substring of an expression e is denoted $\mathsf{subvar}(e)$: $x \in \mathsf{subvar}(e)$ iff $e = x$ or $e = e_1||e_2$ and $x \in \mathsf{subvar}(e_1) \cup \mathsf{subvar}(e_2)$, for some expressions e_1 and e_2.

For example, consider the following expression: $e = (R||(\mathrm{in}_e||\, f(R||r)))||g \oplus G(R)$. Here, $\mathsf{subvar}(e) = \{R, \mathrm{in}_e\}$, but $r, g \notin \mathsf{subvar}(e)$.

Definition 5 Let X be a family of distributions in $\textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ and V_1 and V_2 be sets of local variables or variables in Var. By $D(X, V_1, V_2)$ we denote the following distribution family (on tuples of bit-strings):

$$D(X, V_1, V_2)_\eta = \left[(S, \mathbf{H}, (f,\, f^{-1})) \xleftarrow{r} X_\eta : (S(V_1),\, f(S(V_2)), \mathbf{H},\, f)\right]$$

Here $S(V_1)$ is the point-wise application of S to the elements of V_1 and $f(S(V_2))$ is the point-wise application of f to the elements of $S(V_2)$. We say that X and X' are $V_1; V_2$-*indistinguishable*, denoted by $X \sim_{V_1;V_2} X'$, if $D(X, V_1, V_2) \sim D(X', V_1, V_2)$.

We emphasize that in the above definition, we have that $V_1, V_2 \subseteq \mathsf{Var}$, and since for any $i \in \{1, \ldots, n\}$, $\mathbb{T}_{H_i} \notin \mathsf{Var}$, we get $\mathbb{T}_{H_i} \notin V_1 \cup V_2$ for any $i \in \{1, \ldots, n\}$. Hence, every time we use the equivalence $\sim_{V_1;V_2}$, the variables H_i *are not given to the adversary*.

Example 2 Let S_0 be any state and let H_1 be a hash function. Recall that we are working in the ROM. Consider the following distributions: $X_\eta = [\beta; S := S_0\{x \mapsto u, y \mapsto H_1(u)\} : (S, \mathbf{H}, (f,\, f^{-1}))]$ and $X'_\eta = [\beta; u' \xleftarrow{r} \{0,1\}^\eta; S := S_0\{x \mapsto u, y \mapsto H_1(u')\} : (S, \mathbf{H}, (f,\, f^{-1}))]$, where $\beta = \mathbf{H} \xleftarrow{r} \Omega; (f,\, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); u \xleftarrow{r} \{0,1\}^\eta$. Then, we have $X \sim_{\{y\};\{x\}} X'$ but we do not have $X \sim_{\{y,x\};\emptyset} X'$, because then the adversary can query the value of $H_1(x)$ and match it to that of y.

Definition 6 We write $\nu x \cdot X_\eta$ to denote the following distribution:

$$\left[v \xleftarrow{r} \mathcal{U}_\eta; (S, \mathbf{H}, (f,\, f^{-1})) \xleftarrow{r} X_\eta :\ (S\{x \mapsto v\}, \mathbf{H}, (f,\, f^{-1}))\right],$$

We recall that $\mathcal{U}$ is the family of uniform distributions on the values on which x ranges. We lift this notation to families of distributions in usual way: $\nu x.X = (\nu x.X_\eta)_{\eta\in\mathbb{N}}$.

Our assertion language is defined by the following grammar, where ψ defines the set of atomic assertions:

$$\psi ::= \mathsf{Indis}(\nu x; V_1; V_2) \mid \mathsf{WS}(x; V_1; V_2) \mid \mathsf{H}(H, e)$$
$$\varphi ::= \mathsf{true} \mid \psi \mid \varphi \wedge \varphi,$$

where $V_1, V_2 \subseteq \mathsf{Var}$ and e is an expression constructible (by the adversary) out of the variables used in the program, that is to say, possibly using concatenation, xor, hash oracles or f.

Intuitively, $\mathsf{Indis}(\nu x; V_1; V_2)$ is satisfied by a distribution on configurations, if given values of variables in V_1 and images by f of values of variables in V_2, any polynomial adversary in η has negligible probability to distinguish between the following two distributions: first, the distribution resulting of computations performed using the original value of x as is in X, secondly, the distribution resulting from computations performed replacing everywhere the value of x by a random value of the same length as x. In Section 4.4, in order to analyze schemes using one-way functions f that are not permutations, we generalize the predicate Indis into Indis_f. The predicate Indis_f models the fact that the adversary cannot distinguish between the value of a variable and the image by f of a random value sampled uniformly. The assertion $\mathsf{WS}(x; V_1; V_2)$ stands for Weak Secret and is satisfied by a distribution, if any adversary has negligible probability to compute the value of x, when he is given the values of the variables in V_1 and the image by the one-way permutation of those in V_2. Lastly, $\mathsf{H}(H, e)$ is satisfied when the probability that the value of e has been submitted to the hash oracle H is negligible.

Notations We use $\mathsf{Indis}(\nu x; V)$ instead of $\mathsf{Indis}(\nu x; V; \emptyset)$ and $\mathsf{Indis}(\nu x)$ instead of $\mathsf{Indis}(\nu x; \mathsf{Var})$. Similarly $\mathsf{WS}(x; V)$ stands for $\mathsf{WS}(x; V; \emptyset)$.

Formally, the meaning of the assertion language is defined by a satisfaction relation $X \models \varphi$, which tells us when a family of distributions on configurations X satisfies the assertion φ.

The satisfaction relation $X \models \psi$ is defined as follows:

- $X \models \mathsf{true}$.
- $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \mathsf{Indis}(\nu x; V_1; V_2)$ iff $X \sim_{V_1;V_2} \nu x \cdot X$
- $X \models \mathsf{WS}(x; V_1; V_2)$ iff $\mathsf{Pr}[(S, \mathbf{H}, (f, f^{-1})) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(V_1), f(S(V_2))) = S(x)]$ is negligible, for any adversary $\mathcal{A}^{\mathbf{H}}$.
- $X \models \mathsf{H}(H, e)$ iff $\mathsf{Pr}[(S, \mathbf{H}, (f, f^{-1})) \xleftarrow{r} X_\eta : S(e) \in S(\mathbb{T}_H).\mathsf{dom}]$ is negligible.

Moreover, we write $\varphi \Rightarrow \varphi'$ iff for any family of distributions X such that $X \models \varphi$, then $X \models \varphi'$.

The relation between our Hoare triples and semantic security is established by the following proposition that states that if the value of out_e is indistinguishable from a random value then the scheme considered is IND-CPA (see [2] for details).

Proposition 1 *Let* $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \textbf{var x}; \mathsf{c}, \mathcal{D}(in_d, out_d) : \textbf{var y}; \mathsf{c}')$ *be a generic encryption scheme. It is IND-CPA secure, if* $\{\mathsf{true}\}\mathsf{c}\{\mathsf{Indis}(\nu out_e)\}$ *is valid.*

Indeed, if $\{\mathsf{true}\}\mathsf{c}\{\mathsf{Indis}(\nu \text{out}_e)\}$ holds then the encryption scheme is secure with respect to randomness of ciphertext. It is standard that randomness of ciphertext implies IND-CPA security.

In the rest of the paper, for simplicity, we omit to write the draw of f and its inverse, and we do not mention them in the description of the configurations either.

4.2 Some Properties of the Assertion Language

In this section, we prove properties of our assertions that are useful for proving IND-CPA security of encryption schemes and soundness of our Hoare Logic.

The predicates Indis and WS are compatible with indistinguishability in the following sense:

Lemma 1 *For any* $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$*, any sets of variables* V_1 *and* V_2*, and any variable* x*:*

1. *if* $X \sim_{V_1;V_2} X'$ *then* $X \models \mathsf{Indis}(\nu x; V_1; V_2) \iff X' \models \mathsf{Indis}(\nu x; V_1; V_2)$.
2. *if* $X \sim_{V_1;V_2\cup\{x\}} X'$ *then* $X \models \mathsf{WS}(x; V_1; V_2) \iff X' \models \mathsf{WS}(x; V_1; V_2)$.
3. *if* $X \sim X'$ *then* $X \models \mathsf{H}(H, e) \iff X' \models \mathsf{H}(H, e)$.

Proof By symmetry of indistinguishability and equivalence, for each proposition, the conclusion follows from a single implication.

Let us start with the proof of the first item. We assume $X \sim_{V_1;V_2} X'$ which is equivalent to $X' \sim_{V_1;V_2} X$. Hence, $\nu x.X \sim_{V_1;V_2} \nu x.X'$; this can be justified by an immediate reduction. Moreover, the hypothesis $X \models \mathsf{Indis}(\nu x; V_1; V_2)$ implies $X \sim_{V_1;V_2} \nu x.X$. By transitivity of the indistinguishability relation, we get $X' \sim_{V_1;V_2} \nu x.X'$. Thus, $X' \models \mathsf{Indis}(\nu x; V_1; V_2)$.

We now consider the second item. We prove by reduction that $X' \models \mathsf{WS}(x; V_1; V_2)$ must hold, provided that $X \sim_{V_1;V_2\cup\{x\}} X'$ and $X \models \mathsf{WS}(x; V_1; V_2)$. Given an adversary $\mathcal{A}$ that falsifies $X' \models \mathsf{WS}(x; V_1; V_2)$, we construct an adversary $\mathcal{B}$ that falsifies $X \sim_{V_1;V_2\cup\{x\}} X'$. Informally, on input values for variables in V_1 and V_2, denoted $(\mathbf{v}, \mathbf{v}')$, and a value u for $f(x)$, $\mathcal{B}$ runs A on $(\mathbf{v}, \mathbf{v}')$, which outputs v_x, its guess for x. Then $\mathcal{B}$ compares $f(v_x)$ to u. If the equality holds, $\mathcal{B}$ answers 1 (that is, the values $(\mathbf{v}, \mathbf{v}')$ were sampled according to distribution X'); otherwise $\mathcal{B}$ answers a bit picked at random uniformly.

Let A denote the event $\mathcal{A}^{\mathbf{H}}(S(V_1), f(S(V_2))) = f(S(x))$, $\neg A$ denote the event $\mathcal{A}^{\mathbf{H}}(S(V_1), f(S(V_2))) \neq f(S(x))$ and B denote the event $\mathcal{B}^{\mathbf{H}}(S(V_1), f(S(V_2)),$

$f(S(x))) = 1$. Moreover, let $\mathsf{Adv}_{\mathcal{B}}$ abbreviate $\mathsf{Adv}(\mathcal{B}^{\mathbf{H}}, \eta, D(X, V_1, V_2 \cup \{x\})_\eta, D(X', V_1, V_2 \cup \{x\})_\eta)$. Then, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}} &= \left|\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : B\right] - \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : B\right]\right| \\
&= \left|\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : B|A\right].\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : A\right]\right. \\
&\quad +\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : B|\neg A\right].\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : \neg A\right] \\
&\quad -\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : B|A\right].\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : A\right] \\
&\quad \left.-\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : B|\neg A\right].\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : \neg A\right]\right| \\
&= \left|\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : A\right] + \frac{1}{2}.\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : \neg A\right]\right. \\
&\quad \left.-\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : A\right] - \frac{1}{2}.\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : \neg A\right]\right| \\
&\qquad \text{using } \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : B|A\right] = 1 = \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : B|A\right] \\
&\qquad \text{and } \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : B|\neg A\right] = \frac{1}{2} = \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : B|\neg A\right] \\
&= \frac{1}{2}.\left|\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : A\right] - \frac{1}{2}.\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X : A\right]\right| \\
&\qquad \text{using } \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : \neg A\right] = 1 \quad - \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : A\right] \\
&\qquad \text{and } \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : \neg A\right] = 1 - \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : A\right]
\end{aligned}$$

Therefore,

$$\left|\mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : A\right] - \mathsf{Pr}\left[(S,\mathbf{H}) \xleftarrow{r} X_\eta : A\right]\right| = 2\mathsf{Adv}_{\mathcal{B}}.$$

Since $X \models \mathsf{WS}(x; V_1; V_2)$, we obtain that $\left|\mathsf{Pr}[(S,\mathbf{H}) \xleftarrow{r} X_\eta : A]\right|$ is negligible. Hence $\left|\mathsf{Pr}[(S,\mathbf{H}) \xleftarrow{r} X'_\eta : A]\right|$ is negligible if and only if $\mathsf{Adv}_{\mathcal{B}}$ is negligible.

Concerning the third item, it easy to see that a polynomial adversary dealing with $X \sim X'$, has access to all variables in $\mathsf{Var} \cup \mathbb{T}_{\mathbf{H}}$, and hence he can evaluate the expression e and check whether the value he gets, is or not among the bit-strings obtained by projecting the list given by $\mathbb{T}_H \in \mathbb{T}_{\mathbf{H}}$ to the first element. □

We now present a lemma that relates atomic assertions and states some monotonicity properties:

Lemma 2 *Let $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ be a distribution:*

1. *If $X \models \mathsf{Indis}(\nu x; V_1; V_2)$, $V'_1 \subseteq V_1$ and $V'_2 \subseteq V_1 \cup V_2$ then $X \models \mathsf{Indis}(\nu x; V'_1; V'_2)$.*
2. *If $X \models \mathsf{WS}(x; V_1; V_2)$ and $V'_1 \subseteq V_1$ and $V'_2 \subseteq V_1 \cup V_2$ then $X \models \mathsf{WS}(x; V'_1; V'_2)$.*
3. *If $X \models \mathsf{Indis}(\nu x; V_1; V_2 \cup \{x\})$ and $x \notin V_1 \cup V_2$ then $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$.*

Proof The first two properties are straightforward.
To prove the last assertion, we let $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ such that $X \models \mathsf{Indis}(\nu x; V_1; V_2 \cup \{x\})$. Thus, $X \sim_{V_1; V_2 \cup \{x\}} \nu x.X$. Lemma 1 allows us to say that it is sufficient to prove that $\nu x.X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$ in order to conclude that $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$. Let us consider an adversary $\mathcal{A}$ against $\nu x.X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$. It takes as input, among others, a value $f(u)$ for $f(x)$,

with $u \xleftarrow{r} \mathcal{U}$. If $\mathcal{A}$ successfully computes a value for x, it obviously equals u, so that $\mathcal{A}$ in fact computes the pre-image of the one-way function f on the random value $f(u)$. This latter event has a negligible probability to happen. Hence, we can conclude that $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$. □

An expression e is called *constructible from* $(V_1; V_2)$, if it can be constructed from variables in V_1 and images by f of variables in V_2, calling oracles if necessary. Obviously, we can give an inductive definition: if $x \in V_1$ then x is constructible from $(V_1; V_2)$; if $x \in V_2$ then $f(x)$ is constructible from $(V_1; V_2)$; if e_1, e_2 are constructible from $(V_1; V_2)$ then $H(e_1)$, $f(e_1)$, $e_1||e_2$ and $e_1 \oplus e_2$ are constructible from $(V_1; V_2)$. Then Indis is preserved by constructible computations.

Lemma 3 *For any $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, any sets of variables V_1 and V_2, any expression e constructible from $(V_1; V_2)$, and any variable x, if $X \sim_{V_1;V_2} X'$ then $[\![x := e]\!](X) \sim_{V_1 \cup \{x\};V_2} [\![x := e]\!](X')$.*

Proof We assume $X \sim_{V_1;V_2} X'$. If we suppose that $[\![x := e]\!](X) \not\sim_{V_1 \cup \{x\};V_2} [\![x := e]\!](X')$, then there exists $\mathcal{A}$ a poly-time adversary that, on input V_1, x and $f(V_2)$ drawn either from $[\![x := e]\!](X)$ or $[\![x := e]\!](X')$, guesses the right initial distribution with non-negligible probability.
We let $\mathcal{B}$ be the following adversary against $X \sim_{V_1;V_2} X'$:
$\mathcal{B}(V_1, f(V_2))$:= let $x := e$ in $\mathcal{A}(V_1, x, f(V_2))$.
The idea is that $\mathcal{B}$ can evaluate in polynomial time the expression e using its own inputs. Hence it can provide the appropriate inputs to $\mathcal{A}$. It is clear that the advantage of $\mathcal{B}$ is exactly that of $\mathcal{A}$, which would imply that it is not negligible, although we assumed $X \sim_{V_1;V_2} X'$. □

Corollary 1 *For any $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, any sets of variables V_1 and V_2, any expression e constructible from $(V_1; V_2)$, and any variable x, z such that $z \notin \{x\} \cup \mathsf{Var}(e)$ if $X \models \mathsf{Indis}(\nu z; V_1; V_2)$ then $[\![x := e]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{x\}; V_2)$. We emphasize that here we use the notation $\mathsf{Var}(e)$ (in its usual sense), that is to say, the variable z does not appear* at all *in e.*

Proof $X \models \mathsf{Indis}(\nu z; V_1; V_2)$ is equivalent to $X \sim_{V_1;V_2} \nu z.X$. Using Lemma 3 we get $[\![x := e]\!](X) \sim_{V_1 \cup \{x\};V_2} [\![x := e]\!](\nu z.X)$. Since $z \notin \{x\} \cup \mathsf{Var}(e)$ we have that $[\![x := e]\!](\nu z.X) = \nu z.[\![x := e]\!](X)$ and hence $[\![x := e]\!](X) \sim_{V_1 \cup \{x\};V_2} \nu z.[\![x := e]\!](X)$, that is $[\![x := e]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{x\}; V_2)$. □

Lemma 4 *For any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, any sets of variables V_1 and V_2, any expression e constructible from $(V_1; V_2)$, and any variable $x \neq z$, if $X \models \mathsf{WS}(z; V_1; V_2)$ then $[\![x := e]\!](X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$.*

Proof If we suppose that $[\![x := e]\!](X) \not\models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$, then there exists $\mathcal{A}$ a poly-time adversary that, on input V_1, x and $f(V_2)$ drawn from $[\![x := e]\!](X)$, computes the right value for z with non-negligible probability.
We let $\mathcal{B}$ be the following adversary against $X \models \mathsf{WS}(z; V_1; V_2)$:
$\mathcal{B}(V_1, f(V_2))$:= let $x := e$ in $\mathcal{A}(V_1 \cup \{x\}, f(V_2))$.

Since $\mathcal{A}$ and $\mathcal{B}$ have the same advantage, we obtain a contradiction, so that $[\![x := e]\!](X) \not\models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$ cannot be true. □

4.3 The Hoare Logic

In this section we present our Hoare logic for IND-CPA security. We begin with a set of preservation rules that tell us when an invariant established at the control point before a command can be transferred to the control point following the command. Then, for each command, we present a set of specific rules that allow us to establish new invariants. The commands that are not considered are usually irrelevant for IND-CPA security. We summarize all our Hoare logic rules in Table 3 (given at the end of the paper).

4.3.1 Generic Preservation Rules

We assume $z \neq x$[1] and c is $x \xleftarrow{r} \mathcal{U}$ or of the form $x := e'$ with e' being either $t||y$ or $t \oplus y$ or $f(y)$ or $H(y)$ or $t \oplus H(y)$.

Before getting started, let us notice that for any of these commands c, $[\![\mathsf{c}]\!]$ affects at most x and $\mathbb{T}_H$. The next lemma directly follows from this remark.

Lemma 5 *For every $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, all sets V_1 and V_2 such that $x \notin V_1 \cup V_2$, and all commands c of the form $x \xleftarrow{r} \mathcal{U}$ or $x := e$, we have $[\![c]\!](X) \sim_{V_1;V_2} X$.*

Proof Let $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$.

$$\begin{aligned}
D([\![\mathsf{c}]\!](X), V_1, V_2)_\eta &= D\big(\big[(S, \mathbf{H}) \xleftarrow{r} [\![\mathsf{c}]\!](X_\eta) : (S, \mathbf{H})\big], V_1, V_2\big) \\
&= \big[(S, \mathbf{H}) \xleftarrow{r} X_\eta; (S', \mathbf{H}) \xleftarrow{r} [\![\mathsf{c}]\!]((S, \mathbf{H})) : \\
&\quad \big(S'(V_1),\ f(S'(V_2)), \mathbf{H},\ f\big)\big] \\
&= \big[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \big(S(V_1),\ f(S(V_2)), \mathbf{H},\ f\big)\big] \\
&\quad \text{since } x \notin V_1 \cup V_2 \\
&= D(X, V_1, V_2)_\eta
\end{aligned}$$

□

We now give generic preservation rules for our predicates. We comment them right below.

Lemma 6 *The following rules are sound, when $z \neq x$, and c is $x \xleftarrow{r} \mathcal{U}$ or of the form $x := e'$ with e' being either $t||y$ or $t \oplus y$ or $f(y)$ or $H(y)$ or $t \oplus H(y)$:*

- (G1) $\{\mathit{Indis}(vz; V_1; V_2)\}\ \mathsf{c}\ \{\mathit{Indis}(vz; V_1; V_2)\}$, *provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$.*
- (G2) $\{\mathit{WS}(z; V_1; V_2)\}\ \mathsf{c}\ \{\mathit{WS}(z; V_1; V_2)\}$, *provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$.*

[1] By $x = y$ we mean syntactic equality.

Table 3 Summary of our Hoare logic rules

Generic preservation rules: when $z \neq x$[1] and c is $x \xleftarrow{r} \mathcal{U}$ or of the form $x := e'$ with e' being either $t||y$ or $t \oplus y$ or $f(y)$ or $H(y)$ or $t \oplus H(y)$

(G1) $\{\mathsf{Indis}(\nu z; V_1; V_2)\}\ \mathsf{c}\ \{\mathsf{Indis}(\nu z; V_1; V_2)\}$, provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$

(G2) $\{\mathsf{WS}(z; V_1; V_2)\}\ \mathsf{c}\ \{\mathsf{WS}(z; V_1; V_2)\}$, provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$

(G3) $\{\mathsf{H}(H', e[e'/x])\}\ \mathsf{c}\ \{\mathsf{H}(H', e)\}$, provided $H' \neq H$ in case c is either $x := H(y)$ or $x := t \oplus H(y)$
Here, by convention, $e[e'/x]$ is either e if c is $x \xleftarrow{r} \mathcal{U}$ or the expression obtained from e by replacing x by e' in case $c \equiv x := e'$

Random assignment rules

(R1) $\{\mathsf{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{Indis}(\nu x)\}$

(R2) $\{\mathsf{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{H}(H, e)\}$ if $x \in \mathsf{subvar}(e)$

We assume $x \neq y$, for the next two rules

(R3) $\{\mathsf{Indis}(\nu y; V_1; V_2)\} x \xleftarrow{r} \mathcal{U} \{\mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2)\}$

(R4) $\{\mathsf{WS}(y; V_1; V_2)\} x \xleftarrow{r} \mathcal{U} \{\mathsf{WS}(y; V_1 \cup \{x\}; V_2)\}$

Hash functions rules: when $x \neq y$, and α is either a constant or a variable

(H1) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{H}(H, y)\} x := \alpha \oplus \mathsf{H}(y) \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$

(H2) $\{\mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{H}(H', e)\}$, if $x \in \mathsf{subvar}(e)$

(H3) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2 \cup \{y\})\}$ if $y \notin V_1$

We assume $x \neq y$ and $z \neq x$ for the next rules

(H4) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{WS}(z; V_1; V_2) \wedge \mathsf{H}(H, y)\} x := H(y)\ \{\mathsf{WS}(z; V_1 \cup \{x\}; V_2)\}$

(H5) $\{\mathsf{H}(H, e) \wedge \mathsf{WS}(z; y)\} x := H(y) \{\mathsf{H}(H, e)\}$, if $z \in \mathsf{subvar}(e) \wedge x \notin \mathsf{subvar}(e)$

(H6) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$, if $y \notin V_1$

(H7) $\{\mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z\}; V_2) \wedge \mathsf{H}(H, y)\} x := H(y) \{\mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)\}$

One-way function rules: when $y \notin V \cup \{x\}$

(O1) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathsf{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$

(O2) $\{\mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2 \cup \{y\})\}$, if $z \neq y$

(O3) $\{\mathsf{WS}(z; V_1; V_2) \wedge \mathsf{Indis}(\nu y; V_1; \{y, z\} \cup V_2)\}\ x := f(y)\ \{\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})\}$

(P1) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$, if $y \notin V_1 \cup V_2$

(PO1) $\{\mathsf{Indis}(\nu x; V \cup \{x, y\}) \wedge \mathsf{Indis}(\nu y; V \cup \{x, y\})\} z := f(x||y) \{\mathsf{WS}(x; V \cup \{z\}) \wedge \mathsf{Indis}_f(\nu z; V \cup \{z\})\}$

(P1') $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})\} x := f(y) \{\mathsf{Indis}_f(\nu x; V_1 \cup \{x\}; V_2)\}$ if $y \notin V_1 \cup V_2$

Exclusive or rules: when $y \notin V_1 \cup V_2$, and $y \neq x$

(X1) $\{\mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{Indis}(\nu x; V_1 \cup \{x, z\}; V_2)\}$

(X2) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, provided that $t \neq x, y, z$

(X3) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)\}$, if $t \neq x$

Concatenation rules

(C1) $\{\mathsf{WS}(y; V_1; V_2)\}\ x := y||z\ \{\mathsf{WS}(x; V_1; V_2)\}$, if $x \notin V_1 \cup V_2$. A dual rule applies for z

(C2) $\{\mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2) \wedge \mathsf{Indis}(\nu z; V_1 \cup \{y, z\}; V_2)\}\ x := y||z\ \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$, if $y, z \notin V_1 \cup V_2$

(C3) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\}\ x := y||z\ \{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, if $t \neq x, y, z$

(C4) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\}\ x := y||z\ \{\mathsf{WS}(t; V_1 \cup \{y, z, x\}; V_2)\}$, if $t \neq x$

Consequence and sequential composition rules

(Csq) if $\varphi_0 \Rightarrow \varphi_1$, $\{\varphi_1\}\ \mathsf{c}\ \{\varphi_2\}$ and $\varphi_2 \Rightarrow \varphi_3$ then $\{\varphi_0\}\ \mathsf{c}\ \{\varphi_3\}$

(Seq) if $\{\varphi_0\}\ \mathsf{c}_1\ \{\varphi_1\}$ and $\{\varphi_1\}\ \mathsf{c}_2\ \{\varphi_2\}$, then $\{\varphi_0\}\ \mathsf{c}_1; \mathsf{c}_2\ \{\varphi_2\}$

(Conj) if $\{\varphi_0\}\ \mathsf{c}\ \{\varphi_1\}$ and $\{\varphi_2\}\ \mathsf{c}\ \{\varphi_3\}$, then $\{\varphi_0 \wedge \varphi_2\}\ \mathsf{c}\ \{\varphi_1 \wedge \varphi_3\}$

– (G3) $\{\mathsf{H}(H', e[e'/x])\}\ \mathsf{c}\ \{\mathsf{H}(H', e)\}$, *provided* $H' \neq H$ *in case* c *is either* $x := H(y)$ *or* $x := t \oplus H(y)$. *Here, by convention,* $e[e'/x]$ *is either* e *if* c *is* $x \xleftarrow{r} \mathcal{U}$ *or the expression obtained from* e *by replacing* x *by* e' *in case* $c \equiv x := e'$.

The rules deal with predicates on a variable z different from x, y, t appearing in the command c that is applied. Thus, the predicates Indis and WS are quite intuitively preserved as soon as either x does not appear in sets V_1, V_2 the adversary is provided, or x does appear, but the value of e' was already deducible from values given for V_1, V_2. As for rule (G3), it is meant to express preservation of predicate $\mathsf{H}(H', .)$. Intuitively, if for states drawn in distribution X, $e[e'/x]$ has negligible probability to belong to $\mathbb{T}_H$ before performing $x := e'$, then e has negligible probability to belong to $\mathbb{T}_H$ for states drawn in $[\![x := e']\!](X)$, that is, once the command is performed.

Proof

(G1) The case when e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$ follows from Corollary 1. Let us suppose that $x \notin V_1 \cup V_2$. The previous lemma entails $[\![\mathsf{c}]\!](X) \sim_{V_1;V_2} X$. Then, according to the preservation of properties through indistinguishability proved in Lemma 1, $X \models \mathsf{Indis}(\nu z; V_1; V_2)$ implies $[\![\mathsf{c}]\!](X) \models \mathsf{Indis}(\nu z; V_1; V_2)$. The case of $x \xleftarrow{r} \mathcal{U}$ is obvious.

(G2) As for (G1) using Lemma 4 instead of Corollary 1.

(G3) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ and any $[\![\mathsf{c}]\!]$ affecting at most x and $\mathbb{T}_H$ such that $X \models \mathsf{H}(H', e[e'/x])$. Let $p_\eta = \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} [\![\mathsf{c}]\!](X_\eta) : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$. Then $p_\eta = \mathsf{Pr}[S' \xleftarrow{r} X_\eta; S \xleftarrow{r} [\![\mathsf{c}]\!](S', \mathbf{H}) : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$. Now, $S'(e[e'/x])$[2] is equal to $S(e)$ and $S'(\mathbb{T}_{H'}).\mathsf{dom} = S(\mathbb{T}_{H'}).\mathsf{dom}$. Hence,

$$p_\eta = \mathsf{Pr}\big[S' \xleftarrow{r} X_\eta; S \xleftarrow{r} [\![\mathsf{c}]\!]S' : S'(e[e'/x]) \in S'(\mathbb{T}_{H'}).\mathsf{dom}\big]$$
$$= \mathsf{Pr}\big[(S', \mathbf{H}) \xleftarrow{r} X_\eta : S'(e[e'/x]) \in S'(\mathbb{T}_{H'}).\mathsf{dom}\big]$$

Since $X \models \mathsf{H}(H', e[e'/x])$, the last probability is a negligible function of η. Therefore p_η is also negligible in η and $[\![\mathsf{c}]\!](X) \models \mathsf{H}(H', e)$. □

4.3.2 Random Assignment

Rule (R1) below states that $\mathsf{Indis}(\nu x)$ is satisfied after assigning a randomly sampled value to the variable x. Rule (R2) takes advantage of the fact that the cardinality of $\mathcal{U}$ is exponential in the security parameter, and that since e contains the freshly generated x the probability that it has already been submitted to H is small. Rules (R3) and (R4) state that the value of x cannot help an adversary in distinguishing the value of y from a random value in (R3) or computing its value in (R4). This is the case because the value of x is randomly sampled.

Lemma 7 *The following rules are sound:*

- (R1) $\{\mathit{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathit{Indis}(\nu x)\}$
- (R2) $\{\mathit{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathit{H}(H, e)\}$ *if* $x \in \mathit{subvar}(e)$.

[2] We recall that in the case of $x \xleftarrow{r} \mathcal{U}$, $e[e'/x]$ is actually e.

Additionally, we have the following preservation rules, where we assume $x \neq y$, are sound:

- (R3) $\{\mathit{Indis}(\nu y; V_1; V_2)\}x \xleftarrow{r} \mathcal{U}\{\mathit{Indis}(\nu y; V_1 \cup \{x\}; V_2)\}$
- (R4) $\{\mathit{WS}(y; V_1; V_2)\}x \xleftarrow{r} \mathcal{U}\{\mathit{WS}(y; V_1 \cup \{x\}; V_2)\}$

Proof

(R1) Immediate.

(R2) The fact that $x \in \mathsf{subvar}(e)$ implies that there exists a poly-time function g such that $g(S(e)) = S(x)$ for any state S (namely g consists in extracting the right substring corresponding to x from the expression e). We are interested in bounding

$$\begin{aligned}
&\mathsf{Pr}\big[S \xleftarrow{r} [\![x \xleftarrow{r} \mathcal{U}]\!](X_\eta) : S(e) \in S(\mathbb{T}_H).\mathsf{dom}\big] \\
&\quad = \mathsf{Pr}\big[S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}; S' := S\{x \mapsto u\} : S'(e) \in S'(\mathbb{T}_H).\mathsf{dom}\big] \\
&\quad = \mathsf{Pr}\big[S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}; S' := S\{x \mapsto u\} : S'(e) \in S(\mathbb{T}_H).\mathsf{dom}\big] \\
&\quad = \mathsf{Pr}\big[S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U} : u \in g(S(\mathbb{T}_H).\mathsf{dom})\big]
\end{aligned}$$

which is negligible for the cardinality of $\mathbb{T}_H$ is bounded by a polynomial.

(R3) The intuition is that x being completely random, providing its value to the adversary does not help him in any way. We show the result by reduction. Assume that there exists an adversary B against $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2)$ that can distinguish with non-negligible advantage between y and a random value given the values of $V_1 \cup \{x\}$ and $f(V_2)$. Then, we can construct an adversary $A(V_1, f(V_2))$ playing against $X \models \mathsf{Indis}(\nu y; V_1; V_2)$ that has the same advantage as B: $A(V_1, f(V_2))$ draws a value u at random and runs $B(V_1, u, f(V_2))$, and then returns B's answer. If B has non-negligible advantage, then so does A, which contradicts our hypothesis.

(R4) The previous reduction can be adapted in a straightforward way to prove (R4). □

4.3.3 Hash Functions

In this section, we present a set of proof rules that deal with hash functions in the random oracle model. We first state properties of hash functions that are used to prove soundness of our proof rules.

Preliminary Results

In the random oracle model, hash functions are drawn uniformly at random from the space of functions of suitable type at the beginning of the execution of a program. Thus, the images that the hash function associates to different inputs are completely independent. Therefore, one can delay the draw of each hash value until needed. This is the very idea that the first lemma formalizes. It states that while a hash value has not been queried, then one can redraw it without this changing anything from the adversary's point of view. We introduce the notation $H\{v \mapsto u\}$ to denote the function behaving like H at any point except v, with which it associates u.

Lemma 8 (Dynamic draw) *For any* $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ *and any* $y \in \mathsf{Var}$ *such that* $X \models \mathsf{H}(H, y)$, $X \sim ([(S,H) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; \mathbf{H} \leftarrow \{H\{S(y) \mapsto u\}\} \cup (\mathbf{H} \setminus \{H\}) : (S, \mathbf{H})])_\eta$.

Proof Let $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ and $y \in \mathsf{Var}$ such that $X \models \mathsf{H}(H, y)$. We denote X'_η the distribution $[(S, \mathbf{H}) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; \mathbf{H} \leftarrow \{H\{S(y) \mapsto u\}\} \cup (\mathbf{H} \setminus \{H\}) : (S, \mathbf{H})]$. Let us we recall that for $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, all queries that have been made to the hash oracles are recorded in the lists $\mathbb{T}_H$.

First, we begin with some remarks which help us bounding the advantage of an adversary $\mathcal{A}$ trying to distinguish between X and X'. Since $X \models \mathsf{H}(H, y)$, and using the definition of X', it follows that the probability $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}]$ is equal to $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}] = n(\eta)$ where $n(\cdot)$ is a negligible function. Indeed, the state output by distribution X_η is not modified in the computation of X'_η. Moreover $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}] = \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}]$, since under the condition $S(y) \notin S(\mathbb{T}_H).\mathsf{dom}$, drawing $\mathbf{H}$ in Ω and redrawing the value of H on $S(y)$ yields the same distribution as just drawing $\mathbf{H}$ in Ω.

$$
\begin{aligned}
&\mathsf{Adv}(\mathcal{A}^{\mathbf{H}}, \eta, D(X, \mathsf{Var}, \emptyset)_\eta, D(X', \mathsf{Var}, \emptyset)_\eta) \\
&\quad = \big|\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1] \\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1]\big| \\
&\qquad \text{We then distinguish according to whether } S(y) \in \mathbb{T}_H.\mathsf{dom} \\
&\quad = \big|\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}]. \\
&\qquad \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(y) \notin S(\mathbb{T}_H).\mathsf{dom}] \\
&\qquad + \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}]. \\
&\qquad \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}] \\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}]. \\
&\qquad \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \notin S(\mathbb{T}_H).\mathsf{dom}] \\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}]. \\
&\qquad \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\big| \\
&\qquad \text{We then take into account the equalities between terms justified above.} \\
&\quad = \big|\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}] \\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\big|. \\
&\qquad \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}] \\
&\quad \leq 2.n(\eta)
\end{aligned}
$$

□

We now want to prove something a little stronger, involving the variable $\mathbb{T}_H$. Indeed, to execute the command $x := \alpha \oplus H(y)$, we can either draw a value for $H(y)$ at random and bind it by storing it in $\mathbb{T}_H$, or draw x at random and bind $H(y)$ to be worth $x \oplus \alpha$. This uses the same idea as before, but this time we have to carefully take into account the side effects of the command on $\mathbb{T}_H$. To deal with rebinding matters, we introduce a new notation: $S(\mathbb{T}_H) \bullet (v, u)$ means that if v belongs to $S(\mathbb{T}_H).\mathsf{dom}$, then its associated value in $S(\mathbb{T}_H).\mathsf{res}$ is replaced by u, otherwise, it is the concatenation of (v, u) to $S(\mathbb{T}_H)$.

Definition 7 We define $\mathsf{rebind}_H^{y \mapsto e}(S, \mathbf{H})$ by

$$(S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), S(e))\}, \mathbf{H} \setminus \{H\} \cup \{H\{S(y) \mapsto S(e)\}\}).$$

We extend this definition canonically to any family of distributions $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$:

$\mathsf{rebind}_H^{y \mapsto e}(X) = ([(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathsf{rebind}_H^{y \mapsto e}(S, \mathbf{H})])_\eta$. It simply denotes the family of distributions where $H(S(y))$ is defined to be equal to $S(e)$.

Lemma 9 (Rebinding Lemma) *For any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, any hash function symbol H, any variables x and y, if $X \models \mathsf{H}(H, y)$, then*

$$[\![x := \alpha \oplus H(y)]\!](X) \sim \mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X),$$

where α is either a constant or a variable.

Proof To lighten the proof, we assume without loss of generality that there is only one hash function H. First, since $X \models \mathsf{H}(H, y)$, thanks to the dynamic draw lemma, we know that $X_\eta \sim [(S, H) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta : (S, H\{S(y) \mapsto u\})]$. Then, using a similar reasoning to that used in the proof of Lemma 3 (but this time the adversary $\mathcal{B}$ has to update also the variable T_H and to pass it to the adversary $\mathcal{A}$), we get:

$$[\![x := \alpha \oplus H(y)]\!](X_\eta) \sim [\![x := \alpha \oplus H(y)]\!]\left(\left[(S, H) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta : (S, H\{S(y) \mapsto u\})\right]\right).$$

Executing the hash command, the second distribution is in turn equal to

$$\left[(S, H) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta : (S\{x \mapsto S(\alpha) \oplus u; \mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), u)\}, H\{S(y) \mapsto u\})\right].$$

Then, we eventually replace the draw of y by that of x, and propagate the side effects of that change, to obtain another way to denote the same distribution:

$$\Big[(S, H) \xleftarrow{r} X_\eta; v \xleftarrow{r} \mathcal{U}_\eta : (S\{x \mapsto v, \mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), v \oplus S(\alpha))\}, H\{S(y) \mapsto v \oplus S(\alpha)\})\Big].$$

Now, this last distribution is exactly $(\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X))_\eta$, and we conclude. □

Now we are interested in formally proving the useful and intuitive following lemma, which states that to distinguish between a distribution and its 'rebound' version, an adversary must be able to compute the argument y whose hash value has been rebound. More precisely,

Lemma 10 (Hash vs. rebind) *For any* $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$*, any two variables* x *and* y*, any two finite sets of variables* V_1 *and* V_2*, and any hash function* H*, if* $X \models \mathsf{WS}(y; V_1; V_2)$*, then*

$$X \sim_{V_1;V_2} \mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(X).$$

where α *is either a constant or a variable.*

Proof Consider finite sets V_1 and V_2 and X such that $X \models \mathsf{WS}(y; V_1; V_2)$. The sole difference between the distributions is the value of $H(y)$. Namely,

$$\begin{aligned}
D(\mathsf{rebind}_H^{y \mapsto x \oplus \alpha}(X), V_1, V_2)_\eta = {} & [(S, \mathbf{H}) \xleftarrow{r} X_\eta; \\
& S' \leftarrow S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), S(\alpha \oplus x))\} : \\
& (S'(V_1), f(S'(V_2)), H\{S(y) \mapsto S(\alpha \oplus x)\} \cup (\mathbf{H} \setminus \{H\}))] \\
& \text{since } \mathbb{T}_H \notin V_1 \cup V_2 \text{ by definition,} \\
& \text{and it is the only difference between } S \text{ and } S' \\
= {} & [(S, \mathbf{H}) \xleftarrow{r} X_\eta; \\
& S' \leftarrow S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), S(\alpha \oplus x))\} : \\
& (S(V_1), f(S(V_2)), H\{S(y) \mapsto S(\alpha \oplus x)\} \cup (\mathbf{H} \setminus \{H\}))] \\
& \text{and since } S' \text{ is not used anywhere,} \\
= {} & [(S, \mathbf{H}) \xleftarrow{r} X_\eta : \\
& (S(V_1), f(S(V_2)), H\{S(y) \mapsto S(\alpha \oplus x)\} \cup (\mathbf{H} \setminus \{H\}))]
\end{aligned}$$

An adversary trying to distinguish $D(X, V_1, V_2)_\eta$ from this last distribution can only succeed if it calls H on $S(y)$. However, the probability of an adversary computing $S(y)$ is negligible since $X \models \mathsf{WS}(y; V_1; V_2)$. Therefore, $D(\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(X), V_1, V_2)_\eta \sim D(X, V_1, V_2)_\eta$. □

Proof rules for hash functions

We are now prepared to present and prove our proof rules for hash functions. Rule (H1) captures the main feature of the random oracle model, namely that the hash function is a random function. Hence, if an adversary cannot compute the value of y and this latter has not been hashed yet then he cannot distinguish $H(y)$ from a random value. Rule (H2) is similar to rule (R2): as the hash of a fresh value is seemingly random, it has negligible probability to have already been queried to another hash oracle.

Rule (H3) deserves a more elaborate comment. It concludes to a predicate still involving variable y, which is why it is different from rule (H1). It states that the value of variable x is random given first values of V_1, x and $f(V_2)$ as in rule (H1), but also the value of $f(y)$. Indeed, as the value of y is seemingly random, we can use the definition of one-wayness to state that an adversary cannot efficiently compute a satisfactory value for $f^{-1}(f(y))$. Hence, the value of y is unlikely to be queried to H, and the predicate holds.

Lemma 11 *The following basic rules are sound, when $x \neq y$, and α is either a constant or a variable:*

- (H1) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{H}(H, y)\} x := \alpha \oplus H(y) \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$
- (H2) $\{\mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{H}(H', e)\}$, *if* $x \in \mathsf{subvar}(e)$.
- (H3) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)\}\ \ x := H(y)\ \ \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2 \cup \{y\})\}$ *if* $y \notin V_1$

Proof

(H1) First, we use that $X \models \mathsf{WS}(y; V_1; V_2)$, that provides thanks to rule (R4) $\nu x.X \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2)$. Hence, the 'hash-vs-rebind' Lemma 10 applies, we obtain the following $\nu x.X \sim_{V_1 \cup \{x\}; V_2} \mathsf{rebind}_H^{y \mapsto x \oplus \alpha}(\nu x.X)$. Then, as we assumed that $X \models \mathsf{H}(H, y)$, we can use the rebinding lemma, according to which we have the following $\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X) \sim_{V_1 \cup \{x\}; V_2} [\![x := \alpha \oplus H(y)]\!](X)$. By transitivity of the indistinguishability relation, we thus have $\nu x.X \sim_{V_1 \cup \{x\}; V_2} [\![x := \alpha \oplus H(y)]\!](X)$. Finally, noticing that $\nu x.X \sim_{V_1 \cup \{x\}; V_2} \nu x.[\![x := \alpha \oplus H(y)]\!](X)$ (since carrying out the command only impacts on the values of x and $\mathbb{T}_H$ these family of distributions are in fact equal), we have by transitivity $\nu x.[\![x := \alpha \oplus H(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2} [\![x := \alpha \oplus H(y)]\!](X)$. This last statement is equivalent to $[\![x := \alpha \oplus H(y)]\!](X) \models \mathsf{Indis}(x; V_1 \cup \{x\}; V_2)$.

(H2) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ such that $X \models \mathsf{H}(H, y)$, and let $X' = \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$. Since $X \models \mathsf{H}(H, y)$, the rebinding lemma implies $[\![x := H(y)]\!]X \sim X'$. Consider an expression e such that $x \in \mathsf{subvar}(e)$. Using Lemma 1(3), it suffices to show $X' \models \mathsf{H}(H', e)$, that is, that $p_\eta = \mathsf{Pr}[(S, \mathbf{H}) \overset{r}{\leftarrow} X'_\eta : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$ is negligible.

$$
\begin{aligned}
p_\eta &= \mathsf{Pr}\big[(S, \mathbf{H}) \overset{r}{\leftarrow} \nu x \cdot X_\eta; (S', \mathbf{H}') \leftarrow \mathsf{rebind}_H^{y \mapsto x}(S, \mathbf{H}) : \\
&\qquad S'(e) \in S(\mathbb{T}_H).\mathsf{dom}\big] \\
&\quad \text{since } S'(\mathbb{T}_{H'}).\mathsf{dom} \subseteq S(\mathbb{T}_H).\mathsf{dom} \cup \{S(y)\},\ \text{we have} \\
&\leq \mathsf{Pr}\big[(S, \mathbf{H}) \overset{r}{\leftarrow} \nu x \cdot X_\eta; (S', \mathbf{H}') \leftarrow \mathsf{rebind}_H^{y \mapsto x}(S, \mathbf{H}) : \\
&\qquad S'(e) \in S(\mathbb{T}_H).\mathsf{dom} \text{ or } S'(e) = S(y)\big] \\
&\quad \text{now with } S(e) = S'(e) \text{ by definition of the rebinding:} \\
&= \mathsf{Pr}\big[(S, \mathbf{H}) \overset{r}{\leftarrow} \nu x \cdot X_\eta; (S', \mathbf{H}') \leftarrow \mathsf{rebind}_H^{y \mapsto x}(S, \mathbf{H}) : \\
&\qquad S(e) \in S(\mathbb{T}_H).\mathsf{dom} \text{ or } S(e) = S(y)\big] \\
&\quad \text{we can remove the rebinding, since it does not change the event:} \\
&= \mathsf{Pr}\big[(S, \mathbf{H}) \overset{r}{\leftarrow} \nu x \cdot X_\eta : S(e) \in S(\mathbb{T}_H).\mathsf{dom} \text{ or } S(e) = S(y)\big] \\
&\quad \text{which by definition and because we assume } y \neq x \text{ equals} \\
&= \mathsf{Pr}\big[S_1 \overset{r}{\leftarrow} X_\eta; v \overset{r}{\leftarrow} \mathcal{U}; S \leftarrow S_1\{x \mapsto v\} : S(e) \in S_1(\mathbb{T}_H).\mathsf{dom} \text{ or } S(e) = S_1(y)\big] \\
&\quad \text{and as } x \in \mathsf{subvar}(e),\ \text{i.e. } x \text{ is some substring of } e \\
&\leq \frac{Card(S_1(\mathbb{T}_H).\mathsf{dom}) + 1}{2^{|x|}}
\end{aligned}
$$

Moreover, for every state S_1, $Card(S_1(\mathbb{T}_H).\mathsf{dom})$ is bounded by a polynomial in η, and variables in Var have a size polynomial in η too, so that p_η is indeed a negligible function in η.

(H3) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$. Assume $y \notin V_1$ and $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)$. Then, $X \models \mathsf{WS}(y; V_1; V_2 \cup \{y\})$ follows from the third weakening lemma (see Lemma 2). Consequently, rule H1 provides $[\![x := H(y)]\!](X) \models \mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2 \cup \{y\})$. □

We now comment on four other rules to deal with hash commands which are stated below. The idea behind (H4) is the following one: an adversary that is not able to compute the value of y, can not ask this value to H; hence, the value of x (computed as $H(y)$) seems completely random; so if z was not efficiently computable by the adversary given V_1, $f(V_2)$, it remains so when it is additionally provided x. Rule (H5) states that the fact that the value of e has probably not been hashed yet remains true after a hash command, as long as e contains a variable z whose value is not computable out from y. (H6) and (H7) give necessary conditions to the preservation of indistinguishability that is based on the apparent randomness of a hash value. The intuition behind rule (H6) is very similar to that of rule (H3). As for rule (H7), as we want more than just preserving the seemingly randomness of z with respect to V_1, $f(V_2)$, the conditions under which x doesn't help an adversary are that y is not easily deductible from V_1, V_2 and that x is a fresh hash value.

Lemma 12 *The following preservation rules are sound provided that $x \neq y$ and $z \neq x$:*

- (H4) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{WS}(z; V_1; V_2) \wedge \mathsf{H}(H, y)\} x := H(y) \quad \{\mathsf{WS}(z; V_1 \cup \{x\}; V_2)\}$
- (H5) $\{\mathsf{H}(H, e) \wedge \mathsf{WS}(z; y)\} x := H(y) \{\mathsf{H}(H, e)\}$, *if* $z \in \mathsf{subvar}(e) \wedge x \notin \mathsf{subvar}(e)$
- (H6) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)\} \quad x := H(y) \quad \{\mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$, *if* $y \notin V_1$
- (H7) $\{\mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z\}; V_2) \wedge \mathsf{H}(H, y)\} x := H(y) \{\mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)\}$

Proof

(H4) First, we use rule (R4), to state that since $X \models \mathsf{WS}(y; V_1; V_2)$, $\nu x.X \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2)$. Then, from the hash-vs-rebind Lemma 10 applied on $\nu x.X$, we obtain that $\nu x.X \sim_{V_1 \cup \{x\}; V_2} \mathsf{rebind}_H^{y \mapsto x}(\nu x.X)$. Now, using the assumption $X \models \mathsf{H}(H, y)$ and the rebinding lemma, $\mathsf{rebind}_H^{y \mapsto x}(\nu x.X) \sim_{V_1 \cup \{x\}; V_2} [\![x := H(y)]\!](X)$. Hence, $\nu x.X \sim_{V_1 \cup \{x\}; V_2} [\![x := H(y)]\!](X)$. Besides, as $X \models \mathsf{WS}(z; V_1; V_2)$, rule (R4) provides the conclusion $\nu x.X \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$. With Lemma 1, we can conclude that $[\![x := H(y)]\!](X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$ too.

We could do this proof by reduction too, the main idea being that as the value of x is random to an adversary, any adversary against $\mathsf{WS}(z; V_1; V_2)$ before the execution of the command could simulate an adversary against $\mathsf{WS}(z; V_1 \cup \{x\}; V_2)$ by providing this latter with a randomly sampled value in place of x. Both those adversaries would therefore have the same advantage.

(H5) Since $z \in \mathsf{subvar}(e)$, there is a polynomial function g such that for every S, $g(S(e)) = S(z)$ (namely g consists in extracting the right substring

corresponding to z from the expression e). Given $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, let p_η be equal to:

$$\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta; (S', \mathbf{H}') \xleftarrow{r} [\![x := H(y)]\!](S) : S'(e) \in S'(\mathbb{T}_H).\mathsf{dom}]$$

Then, since the command only has an effect on x and $\mathbb{T}_H$,

$$\begin{aligned} p_\eta &= \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta; (S', \mathbf{H}') \xleftarrow{r} [\![x := H(y)]\!](S) : \\ &\qquad S(e) \in S(\mathbb{T}_H).\mathsf{dom} \cup \{S(y)\}] \\ &\leq \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) \in S(\mathbb{T}_H)] + \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) = S(y)] \end{aligned}$$

Now, we can bound the second term as follows:

$$\begin{aligned} \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) = S(y)] &\leq \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : g(S(e)) = g(S(y))] \\ &= \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(z) = g(S(y))] \\ &\qquad \text{for this is how } g \text{ was defined} \end{aligned}$$

Besides, $X \models \mathsf{WS}(z; y)$, so that the probability one can extract the value of z from that of y is negligible. Moreover if $X \models \mathsf{H}(H, e)$ then $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) \in S(\mathbb{T}_H)]$ is negligible.

(H6) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$. Assume $y \notin V_1$, and $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)$. By rule (R3) for random assignment, since $y \neq x$, $\nu x \cdot X \models \mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})$. Therefore, using that $y \notin V_1$ and $y \neq x$ and applying Lemma 2.3 we get $\nu x \cdot X \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})$. Now, the hash-vs-rebind Lemma 10 provides us with $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \sim_{V_1 \cup \{x\}; V_2 \cup \{y\}} \nu x \cdot X$. Thus, $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \models \mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})$ by Lemma 1. Since $X \models \mathsf{H}(H, y)$, by the rebinding lemma, we have $[\![x := H(y)]\!] \sim \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$, so that the result follows from applying once more Lemma 1.

(H7) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ such that $X \models \mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z\}; V_2) \wedge \mathsf{H}(H, y)$. By rule (R3) and (R4) for random assignment, and because $x \neq z$, y, $\nu x \cdot X \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z, x\}; V_2)$. Therefore, the hash-vs-rebind Lemma 10 allows to conclude that $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \sim_{V_1 \cup \{z,x\}; V_2} \nu x \cdot X$. Thus, by the preservation Lemma 1, $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)$. Finally, the rebinding lemma entails $[\![x := H(y)]\!](X) \sim \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$. Therefore $[\![x := H(y)]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)$, once more by Lemma 1.

□

4.3.4 One-Way Functions

Rules to deal with one-way functions are given below. The first rule captures one-wayness of f. Indeed, it states that an adversary can not efficiently compute a pre-image to an apparently random challenge. Rule (O2) and (O3) are meant to provide a little more than mere preservation of the properties of z. (O2) is quite obvious since $f(y)$ is given to the adversary in the precondition. As for rule (O3), it follows from the fact that since y is apparently random with respect to values V_1, z, $f(V_2)$, hence computing x boils down to computing the image by f of a random value.

Consequently, providing an adversary with the values of x and $f(y)$ does not help it. Rule (P1) simply ensues from the fact that f is a permutation and is thus surjective. However, y has to be removed from the sets in the conclusion, otherwise an adversary could compare the value of $f(y)$ with the value given for x and trivially tell if x is real or random.

Lemma 13 *The following rules are sound when $z \neq x$:*

- (O1) $\{\mathit{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathit{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$ *if* $y \notin V_1 \cup \{x\}$.
- (O2) $\{\mathit{Indis}(\nu z; V_1 \cup \{z\}; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathit{Indis}(\nu z; V_1 \cup \{z, x\}; V_2 \cup \{y\})\}$, *if* $z \neq y$
- (O3) $\{\mathit{WS}(z; V_1; V_2) \wedge \mathit{Indis}(\nu y; V_1; \{y, z\} \cup V_2)\}\ x := f(y)\ \{\mathit{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})\}$

For one-way permutations, we also have the following rule:

- (P1) $\{\mathit{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathit{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$, *if* $y \notin V_1 \cup V_2$

Proof

(O1) Let X be such that $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})$. It follows from Lemma 2 that $X \models \mathsf{WS}(y; V_1; V_2 \cup \{y\})$. Since $f(y)$ is obviously constructible from $(V_1; V_2 \cup \{y\})$, we apply Lemma 4, to obtain $[\![x := f(y)]\!](X) \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})$. Notice that the one-wayness of f is not used apparently here. Indeed, the proof of the weakening lemma (see Lemma 2) uses it, and once we apply it, there is only a simple rewriting step left to be able to conclude.

(O2) Since $f(y)$ is constructible from $(V_1 \cup \{z\}; V_2 \cup \{y\})$, we apply Corollary 1 to obtain $[\![x := f(y)]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2 \cup \{y\})$.

(O3) If $z = y$, then the assertion is a consequence of Rule (O1). Hence, we assume $z \neq y$. From $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{z, y\})$ it follows by definition that $X \sim_{V_1; V_2 \cup \{z,y\}} \nu y.X$. Using Lemma 3 we get $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2 \cup \{z,y\}} [\![x := f(y)]\!](\nu y.X)$. Now using Lemma 1, to be able to conclude to $[\![x := f(y)]\!](X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$, it suffices to prove $[\![x := f(y)]\!](\nu y.X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$. Intuitively, this comes from the randomness of x and y, which allows us to think it is useless to any adversary trying to compute z. Formally, we show that: $\mathsf{Pr}[S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; S_1 = S\{y \mapsto u; x \mapsto f(u)\} : \mathcal{A}(S_1(V_1), S_1(x), f(S_1(V_2)), f(S_1(y))) = S_1(z)]$ is negligible. Now let $\mathcal{A}$ be an efficient adversary against $\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$. Let $\mathcal{B}(\mathbf{v})$ be the adversary against $\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$ that proceeds as follows: it samples a value $u \xleftarrow{r} \mathcal{U}_\eta$ and replaces every occurrence of y by u, and every occurrence of x by $f(u)$, in the values $\mathbf{v}$ it got as an input. This provides a tuple of values $\mathbf{v}'$. Adversary $\mathcal{B}$ runs $\mathcal{A}$ on $\mathbf{v}'$, before outputting $\mathcal{A}$'s guess for the value of z. This adversary $\mathcal{B}$ has the same advantage as $\mathcal{A}$ in falsifying $\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$. As we assumed this latter was an efficient adversary, $\mathcal{B}$ is efficient as well, which contradicts $X \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$.

(P1) Since $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})$ and $f(y)$ is constructible from $(V_1; V_2 \cup \{y\})$, we apply Lemma 3 to obtain $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2 \cup \{y\}} [\![x := f(y)]\!](\nu y.X)$, and by weakening (see Lemma 2) we get $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2}$

$[\![x := f(y)]\!](\nu y.X)$. Using that f is a permutation and $y \notin V_1 \cup V_2$, we have $D([\![x := f(y)]\!](\nu y.X), V_1 \cup \{x\}, V_2) = D(\nu x.X, V_1 \cup \{x\}, V_2)$, and hence by transitivity of indistinguishability, $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2} \nu x.X$. Now we use $\nu x.X = \nu x.[\![x := f(y)]\!](X)$ to conclude.

□

4.3.5 The Exclusive or Operator

In the following rules, we assume $y \neq z$. To understand rule (X1) one should consider y as a key and think about x as the one-time pad encryption of z with the key y. Of course, y has to be random given y *and* z and not just only y; otherwise, there may exist a some relation between both subterms of x that may allow an adversary to distinguish this latter from a random value. Rules (X2) and (X3) take advantage of the fact that is easy to compute x given y and z.

Lemma 14 *The following rule is sound when $y \notin V_1 \cup V_2$, and $y \neq x$:*

– (X1) $\{\mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2)\}x := y \oplus z\{\mathsf{Indis}(\nu x; V_1 \cup \{x, z\}; V_2)\}$,

Moreover, we have the following rules that are sound:

– (X2) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\}x := y \oplus z\{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, *provided that $t \neq x, y, z$.*
– (X3) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\}x := y \oplus z\{\mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)\}$, *if $t \neq x$.*

Proof

(X1) Let X be such that $X \models \mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2)$, which means $X \sim_{(V_1 \cup \{y,z\}; V_2)} \nu y.X$. Moreover, $y \oplus z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$. We apply Lemma 3 to obtain $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,y,z\}; V_2} [\![x := y \oplus z]\!](\nu y.X)$, and by weakening (see Lemma 2) it we get $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,z\}; V_2} [\![x := y \oplus z]\!](\nu y.X)$.

$$
\begin{aligned}
& D([\![x := y \oplus z]\!](\nu y.X), V_1 \cup \{x, z\}, V_2)_\eta \\
& \quad = [S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; S' := S\{y \mapsto u\}; S'' \xleftarrow{r} [\![x := y \oplus z]\!](S') : \\
& \quad\quad S''(V_1 \cup \{x, z\}),\ f(S''(V_2))] \\
& \quad = [S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; S' := S\{y \mapsto u\}; S'' := S'\{x \mapsto u \oplus S(z)\} : \\
& \quad\quad S''(V_1 \cup \{x, z\}),\ f(S''(V_2))]
\end{aligned}
$$

and since xor is a permutation we can write:

$$
\begin{aligned}
& = [S \xleftarrow{r} X_\eta; v \xleftarrow{r} \mathcal{U}_\eta; S'' := S\{x \mapsto v; y \mapsto v \oplus S(z)\} : \\
& \quad S''(V_1 \cup \{x, z\}),\ f(S''(V_2))] \\
& \quad \text{but changing } y \text{ is useless since } y \notin V_1 \cup V_2 \cup \{z\} \\
& = [S \xleftarrow{r} X_\eta; v \xleftarrow{r} \mathcal{U}_\eta; S'' := S\{x \mapsto v\} : S''(V_1 \cup \{x, z\}),\ f(S''(V_2))] \\
& = D(\nu x.X, V_1 \cup \{x, z\}, V_2)_\eta
\end{aligned}
$$

From this equality of distributions, we get $[\![x := y \oplus z]\!](\nu y.X) \sim_{V_1 \cup \{x,z\}; V_2} \nu x.X$. Then, by transitivity of indistinguishability, we can conclude that $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,z\}; V_2} \nu x.X$. Then, as $\nu x.X = \nu x.[\![x := y \oplus z]\!](X)$, and applying transitivity once more, we can conclude to $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,z\}; V_2} \nu x.[\![x := y \oplus z]\!](X)$, which is exactly the definition of $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu x; V_1 \cup \{x, z\}; V_2)$.

(X2) Since $y \oplus z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Corollary 1 to obtain $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)$.

(X3) Since $y \oplus z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Lemma 4 to obtain $[\![x := y \oplus z]\!](X) \models \mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)$.

□

4.3.6 Concatenation

We have four rules to deal with concatenation command $x := y||z$. Rule (C1) states that if computing a substring of x out of the elements of V_1 and V_2 is hard, then so is computing x itself. The idea behind (C2) is that y and z being random implies randomness of x, with respect to V_1 and V_2. Of course, y has to be random given y *and* z and not just only y; otherwise, there might exist a dependency between both substrings of x that allows an adversary to distinguish this latter from a random value. A similar comment can be made concerning z. Eventually, rules (C3) and (C4) are more than the simple preservation of the properties of a variable t different from x, y, z that the preservation rules would provide. The value of x being easily computable from those of y and z accounts for soundness of these rules.

Lemma 15 *The following rules are sound:*

- (C1) $\{\mathsf{WS}(y; V_1; V_2)\}\ x := y||z\ \{\mathsf{WS}(x; V_1; V_2)\}$, *if* $x \notin V_1 \cup V_2$. *A dual rule applies for* z.
- (C2) $\{\mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2) \wedge \mathsf{Indis}(\nu z; V_1 \cup \{y, z\}; V_2)\}\ x := y||z\ \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$, *if* $y, z \notin V_1 \cup V_2$
- (C3) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\}\ x := y||z\ \{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, *if* $t \neq x, y, z$
- (C4) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\}\ x := y||z\ \{\mathsf{WS}(t; V_1 \cup \{y, z, x\}; V_2)\}$, *if* $t \neq x$

Proof

(C1) From $X \models \mathsf{WS}(y; V_1; V_2)$, for any adversary $\mathcal{A}$, we have that the probability $\mathsf{Pr}[S \xleftarrow{r} X_\eta : \mathcal{A}(S(V_1), f(S(V_2))) = S(y)]$ is negligible. This implies that for any adversary $\mathcal{B}$, $\mathsf{Pr}[S \xleftarrow{r} X_\eta : \mathcal{B}(S(V_1), f(S(V_2))) = S(y)||S(z)]$ is negligible; otherwise we can build an adversary $\mathcal{A}$ that uses $\mathcal{B}$ as a subroutine and whose advantage is the same as $\mathcal{B}$'s advantage as follows. $\mathcal{A}$ calls $\mathcal{B}$ and then uses the answer of $\mathcal{B}$ to extract the value of $S(y)$ from $S(y)||S(z)$. Since $x \notin V_1 \cup V_2$, we get that for any adversary $\mathcal{B}$, $\mathsf{Pr}[S \xleftarrow{r} [\![x := y||z]\!](X_\eta) : \mathcal{B}(S(V_1), f(S(V_2))) = S(x)] = \mathsf{Pr}[S \xleftarrow{r} X_\eta : \mathcal{B}(S(V_1), f(S(V_2))) = S(y)||S(z)]$, which is negligible.

(C2) We have that $X \models \mathsf{Indis}(\nu z; V_1 \cup \{y, z\}; V_2) \Rightarrow X \sim_{V_1 \cup \{y,z\}; V_2} \nu z.X$, so that in turn $\nu y.X \sim_{V_1 \cup \{y,z\}; V_2} \nu y.\nu z.X$. But $X \models \mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2)$ can be written as $X \sim_{V_1 \cup \{y,z\}; V_2} \nu y.X$. Hence, by transitivity we get $X \sim_{V_1 \cup \{y,z\}; V_2} \nu y.\nu z.X$. Since $y||z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Lemma 3

to obtain $[\![x := y||z]\!](X) \sim_{V_1 \cup \{x,y,z\}; V_2} [\![x := y||z]\!](\nu y.\nu z.X)$, and by weakening (see Lemma 2) we get $[\![x := y||z]\!](X) \sim_{V_1 \cup \{x\}; V_2} [\![x := y||z]\!](\nu y.\nu z.X)$. Using the properties of $||$ and that $\{y, z\} \cap (V_1 \cup V_2) = \emptyset$, we have $D([\![x := y||z]\!](\nu y.\nu z.X), V_1 \cup \{x\}, V_2) = D(\nu x.X, V_1 \cup \{x\}, V_2)$, and hence by transitivity of indistinguishability, $[\![x := y||z]\!](X) \sim_{V_1 \cup \{x\}; V_2} \nu x.X$.

(C3) Since $y||z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Corollary 1 to obtain $[\![x := y||z]\!](X) \models \mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)$.

(C4) Since $y||z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Lemma 4 to obtain $[\![x := y||z]\!](X) \models \mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)$.

□

4.3.7 Additional General Rules

Classically, to reason on programs built according to the language grammar described in Table 1, we additionally need the following couple of rules.

Lemma 16 *Let $\varphi_0, \varphi_1, \varphi_2, \varphi_3$ be assertions from our language, and $\mathsf{c}, \mathsf{c}_1, \mathsf{c}_2$ be any commands. The following rules are sound:*

- (Csq) *if* $\varphi_0 \Rightarrow \varphi_1$ *and* $\{\varphi_1\}\ \mathsf{c}\ \{\varphi_2\}$ *and* $\varphi_2 \Rightarrow \varphi_3$ *then* $\{\varphi_0\}\ \mathsf{c}\ \{\varphi_3\}$.
- (Seq) *if* $\{\varphi_0\}\ \mathsf{c}_1\ \{\varphi_1\}$ *and* $\{\varphi_1\}\ \mathsf{c}_2\ \{\varphi_2\}$ *then* $\{\varphi_0\}\ \mathsf{c}_1; \mathsf{c}_2\ \{\varphi_2\}$.
- (Conj) *if* $\{\varphi_0\}\ \mathsf{c}\ \{\varphi_1\}$ *and* $\{\varphi_0\}\ \mathsf{c}\ \{\varphi_2\}$, *then* $\{\varphi_0\}\ \mathsf{c}\ \{\varphi_1 \wedge \varphi_2\}$.

We omit the proofs of these classical rules. The soundness of the Hoare logic follows by induction from the soundness of each rule.

Proposition 2 *The Hoare triples given in Section 4.3 are valid.*

Example 3 We illustrate our proposition with Bellare & Rogaway's generic construction [7], which can be written shortly as $f(r)||(\mathrm{in}_e \oplus G(r))||H(\mathrm{in}_e||r)$. Where we note $\mathsf{Var} = \{\mathrm{in}_e, \mathrm{out}_e, x_\sigma, r, a, g, b, s, c\}$.

var $r; a; g; b; s; c;$
true
1. $r \xleftarrow{r} \{0,1\}^{n_0}$ using $(R1)$, $(R2)$, and $(R2)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
2. $a := f(r)$ using $(P1)$, $(O1)$, $(G3)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{WS}(r; \mathsf{Var} \setminus \{r\}) \wedge$
$\mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
3. $g := G(r)$ using $(H7)$, $(H1)$, $(H4)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{Indis}(\nu g; \mathsf{Var} \setminus \{r\}) \wedge$
$\mathsf{WS}(r; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
4. $b := \mathrm{in}_e \oplus g$ using $(X2)$, $(X1)$, $(X3)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{g, r\}) \wedge$
$\mathsf{WS}(r; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
5. $s := \mathrm{in}_e||r$ using $(G1)$, $(G1)$, $(C1)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r, s\}) \wedge$
$\mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{g, r, s\}) \wedge$
$\mathsf{WS}(s; \mathsf{Var} \setminus \{r, s\}) \wedge \mathsf{H}(H, s)$

6. $c := H(s)$ using $(H7)$, $(H7)$, and $(H1)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r, s\}) \wedge$
$\mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{r, g, s\}) \wedge$
$\mathsf{Indis}(\nu c; \mathsf{Var} \setminus \{r, s\})$
7. $\text{out}_e := a||b||c$ using $(C2)$ twice
$\mathsf{Indis}(\nu \text{out}_e; \mathsf{Var} \setminus \{a, b, c, r, g, s\})$

4.4 Extensions of the Logic

In this section, we show how our Hoare logic, and hence our verification procedure, can be adapted to deal with injective partially trapdoor one-way functions. This extension is motivated by Pointcheval's construction in [19].

The first observation we have to make is that Proposition 1 is too demanding in case f is not a permutation. Therefore, we introduce a new predicate $\mathsf{Indis}_f(\nu x; V_1; V_2)$ whose meaning is as follows:

$X \models \mathsf{Indis}_f(\nu x; V_1; V_2)$ if and only if $X \sim_{V_1;V_2} [u \xleftarrow{r} \mathcal{U}; (S, \mathbf{H}) \xleftarrow{r} X : (S\{x \mapsto f(u)\}, \mathbf{H})]$.

Notice that, when f is a bijection, $\mathsf{Indis}_f(\nu x; V_1; V_2)$ is equivalent to $\mathsf{Indis}(\nu x; V_1; V_2)$ (f can be the identity function as in the last step of Example 4. Now, let out_e, the output of the encryption oracle, have the form $a_1||\cdots||a_n$ with $a_i = f_i(x_i)$, where f_i are arbitrary functions. Then, we can prove the following:

Proposition 3 *Let GE be a generic encryption scheme of the form* $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c})$, *and let* f_i *be any functions. Let assume that* out_e, *the output of the encryption oracle, has the form* $a_1||\cdots||a_n$ *with* $a_i = f_i(x_i)$.

If $\{\mathsf{true}\}\mathsf{c}\{\bigwedge_{i=1}^{n} \mathsf{Indis}_{f_i}(\nu a_i; a_1, \ldots, a_n, \mathsf{Var} \setminus \{out_e\})\}$ *is valid then GE is IND-CPA.*

The proof of this proposition follows from the transitivity of the relation $\sim_{V_1;V_2}$. Now, we introduce a new rule for $\mathsf{Indis}_f(\nu x; V_1; V_2)$ that replaces rule (P1) in case the one-way function f is not a permutation:

(P1') $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathsf{Indis}_f(\nu x; V_1 \cup \{x\}; V_2)\}$ if $y \notin V_1 \cup V_2$.

Many of rules that hold for *Indis* could be generalized to Indis_f. For simplicity we consider only the rules that are needed in the examples. Clearly all preservation rules can be generalized for Indis_f. Concretely, we have the following lemma, and the proof is similar as for the case Indis.

Lemma 17 (Generalization to Indis_f) *Let* $X, X' \in \text{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ *be arbitrary distributions, let* V_1 *and* V_2 *be arbitrary sets of variables, and let* x, y, z, t *be arbitrary variables. Then, the following assertions hold.*

1. *If* $X \sim_{V_1;V_2} X'$ *then* $X \models \mathsf{Indis}_f(\nu x; V_1; V_2) \iff X' \models \mathsf{Indis}_f(\nu x; V_1; V_2)$.
2. *For any expression e constructible from* $(V_1; V_2)$ *such that* $z \notin \{x\} \cup \mathsf{Var}(e)$, *if* $X \models \mathsf{Indis}_f(\nu z; V_1; V_2)$ *then* $[\![x := e]\!](X) \models \mathsf{Indis}_f(\nu z; V_1 \cup \{x\}; V_2)$.
3. *If* $z \neq x$, *and* c *is* $x \xleftarrow{r} \mathcal{U}$ *or* $x := e'$ *with* $e' \in \{t||y, t \oplus y, f(y), H(y), t \oplus H(y)\}$, *then*
$(\text{G1})^f\ \{\mathsf{Indis}_f(\nu z; V_1; V_2)\}\ \mathsf{c}\ \{\mathsf{Indis}_f(\nu z; V_1; V_2)\}$,
provided that $x \notin V_1 \cup V_2$ *or* e' *is constructible from* $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$.

4. $(\mathrm{R3})^f$ $\{\mathsf{Indis}_f(\nu y; V_1; V_2)\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{Indis}_f(\nu y; V_1 \cup \{x\}; V_2)\}$, *provided* $x \neq y$.
5. $(\mathrm{H7})^f$ $\{\mathsf{Indis}_f(\nu z; V_1 \cup \{z\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z\}; V_2) \wedge \mathsf{H}(H, y)\}\ x := H(y)$ $\{\mathsf{Indis}_f(\nu z; V_1 \cup \{z, x\}; V_2)\}$, *provided that* $x \neq y$ *and* $z \neq x$.
6. $(\mathrm{X2})^f$ $\{\mathsf{Indis}_f(\nu t; V_1 \cup \{y, z\}; V_2)\}x := y \oplus z\{\mathsf{Indis}_f(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, *provided that* $t \neq x, y, z$.

The reader should notice that some rules that hold for Indis can not be generalized to Indis_f. It is the case for (P1), (X1), (C2), etc.

Injective Partially Trapdoor One-way Functions In contrast to the previous section, we do not assume f to be a permutation. On the other hand, we demand a stronger property than one-wayness. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function and let $f^{-1} : \mathcal{Z} \to \mathcal{X}$ be such that $\forall z \in \mathsf{dom}(f^{-1}) \exists y \in \mathcal{Y},\ z = f(f^{-1}(z), y)$. Here f^{-1} is a partial function. The function f is said *partially one-way*, if for any given $z = f(x, y)$, it is computationally impossible to compute a corresponding x. In order to deal with the fact that f is now partially one-way, we add the following rules, where we assume $x, y \notin V \cup \{z\}$ and where we identify f and $(x, y) \mapsto f(x||y)$:

$$(\mathrm{PO1})\{\mathsf{Indis}(\nu x; V \cup \{x, y\}) \wedge \mathsf{Indis}(\nu y; V \cup \{x, y\})\}z := f(x||y)\ \{\mathsf{Indis}_f(\nu z; V \cup \{z\}) \wedge \mathsf{WS}(x; V \cup \{z\})\}$$

The intuition behind the first part of (PO1) is that f guarantees one-way secrecy of the x-part of $x||y$. The second part follows the same idea that (P1').

Example 4 We verify Pointcheval's transformer [19], which can be written shortly as $f(r||H(\mathrm{in}_e||s))||(\mathrm{in}_e||s) \oplus G(r)$. We note $\mathsf{Var} = \{\mathrm{in}_e, \mathrm{out}_e, x_\sigma, r, s, w, h, a, b\}$.

var $r; s; w; h; a; b$;
true

1. $r \xleftarrow{r} \{0,1\}^{n_0}$	using $(R1)$ and $(R2)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r)$	
2. $s \xleftarrow{r} \{0,1\}^{n_0}$	using $(R3)$, $(R1)$, $(G3)$ and $(R2)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu s; \mathsf{Var}) \wedge$ $\mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e\|\|s)$	
3. $w := \mathrm{in}_e\|\|s$	using $(C3)$, $(C1)$, $(G3)$, and $(G3)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{WS}(w; \mathsf{Var} \setminus \{s, w\}) \wedge$ $\mathsf{H}(G, r) \wedge \mathsf{H}(H, w)$	
4. $h := H(w)$	using $(H7)$, $(H1)$, and $(G3)$
$\mathsf{Indis}(\nu r; \mathsf{Var} \setminus \{w, s\}) \wedge$ $\mathsf{Indis}(\nu h; \mathsf{Var} \setminus \{w, s\}) \wedge \mathsf{H}(G, r)$	
5. $a := f(r\|\|h)$	using the new rule $(PO1)$ and $(G3)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} \setminus \{r, s, w, h\}) \wedge$ $\mathsf{WS}(r; \mathsf{Var} \setminus \{r, s, w, h\}) \wedge \mathsf{H}(G, r)$	
6. $b := w \oplus G(r)$	using $(G1)^f$ and $(H1)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} \setminus \{r, s, w, h\}) \wedge$ $\mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{r, s, w, h\})$	
By the Consequence rule using (†)	
$\mathsf{Indis}_f(\nu a; a, b, \mathsf{Var} \setminus \{r, s, w, h, \mathrm{out}_e\}) \wedge$ $\mathsf{Indis}(\nu b; a, b, \mathsf{Var} \setminus \{r, s, w, h, \mathrm{out}_e\})$	

7. $\text{out}_e := a||b$
$\mathsf{Indis}_f(\nu a; a, b, \mathsf{Var} \setminus \{r, s, w, h, \text{out}_e\}) \wedge$ using $(G1)^f$ and $(G1)$
$\mathsf{Indis}(\nu b; a, b, \mathsf{Var} \setminus \{r, s, w, h, \text{out}_e\})$

(†) $\mathsf{Indis}_f(\nu a; a, V) \wedge \mathsf{Indis}(\nu b; a, b, V)$ implies $\mathsf{Indis}_f(\nu a; a, b, V \setminus \{x\})$

5 Automation

We can now fully automate our verification procedure of IND-CPA for the encryption schemes. The idea is, for a given program, to compute invariants backwards, starting with the invariant $\mathsf{Indis}(\nu \text{out}_e; \text{out}_e, \text{in}_e, x_\sigma)$ at the end of the program.

As several rules can lead to the same postcondition, we in fact compute a set of sufficient conditions at all points of the program: for each set (of postconditions) $\{\phi_1, \ldots, \phi_n\}$ and each instruction c, we can compute a set of assertions (preconditions) $\{\phi'_1, \ldots, \phi'_m\}$ such that, for each $i = 1, \ldots, n$, there exists a subset $J \subseteq [1, \ldots, m]$ such that $\{\bigwedge_{j \in J} \phi'_j\} c \{\phi_i\}$ can be derived using the rules given Section 4.3,

The set $\{\phi'_1, \ldots, \phi'_m\}$ is computed by applying two steps:

1. First, a set of assertions are computed by matching the command and assertion ϕ_i with postconditions of the Hoare axioms. This allows to compute for each assertion ϕ_i a set of preconditions $\textsc{Pre}(\phi_i)$.
2. Next, the consequence rule is applied using Lemma 2, i.e., the assertions in $\textsc{Pre}(\phi_i)$ are replaced by stronger assertions, leading to the assertions in $\{\phi'_1, \ldots, \phi'_m\}$.

Since the commands we consider do not include loops, our verification procedure always terminates. However, this verification is potentially exponential in the number of instructions in the encryption command as each postcondition may potentially have several preconditions. This does not seem to be a problem in practice. Indeed, checking Bellare & Rogaway generic construction, for instance, is instantaneous. We implemented that procedure as an Objective Caml program, taking as input a representation of the encryption program.

6 Conclusion

In this paper we proposed an automatic method to prove IND-CPA security of generic encryption schemes in the random oracle model. Then, IND-CCA can be proved using a general method for proving plaintext awareness as described in [11]. It does not seem difficult to adapt our Hoare logic to allow a security proof in the concrete framework of provable security. Another extension of our Hoare logic could concern OAEP. Here, we need to express that the value of a given variable is indistinguishable from a random value as long as a value r has not been submitted to a hash oracle G. This can be done by extending the predicate $\mathsf{Indis}(\nu x; V_1; V_2)$. The details are future work.

Acknowledgements We thank the anonymous and non-anonymous reviewers for their valuable comments that greatly helped improving the paper.

References

1. Barthe, G., Cederquist, J., Tarento, S.: A machine-checked formalization of the generic model and the random oracle model. In: Basin, D., Rusinowitch, M. (eds.) Proceedings of IJCAR'04, vol. 3097 of LNCS, pp. 385–399 (2004)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS, pp. 394–403 (1997)
3. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology, pp. 26–45, London, UK. Springer, Heidelberg (1998)
4. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: POPL '09: Proceedings of the 36th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 90–101. ACM, New York (2009)
5. Blanchet, B.: A computationally sound mechanized prover for security protocols. In: IEEE Symposium on Security and Privacy (S&P 2006), 21–24, pp. 140–154. IEEE Computer Society, Washington (2006)
6. Blanchet, B., Pointcheval, D.: Automated security proofs with sequences of games. In: Dwork, C. (ed.) CRYPTO, vol. 4117 of Lecture Notes in Computer Science, pp. 537–554. Springer, Heidelberg (2006)
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS '93: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM, New York (1993)
8. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT, vol. 950 of Lecture Notes in Computer Science, pp. 92–111. Springer, Heidelberg (1994)
9. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331. http://eprint.iacr.org/ (2004)
10. Barthe, G., Tarento, S.: A machine-checked formalization of the random oracle model. In: Filliâtre, J.-C., Paulin-Mohring, C., Werner, B. (eds.) Proceedings of TYPES'04, vol. 3839 of Lecture Notes in Computer Science, pp. 33–49. Springer, Heidelberg (2004)
11. Courant, J., Daubignard, M., Ene, C., Lafourcade, P., Lahknech, Y.: Towards automated proofs for asymmetric encryption schemes in the random oracle model. Technical report, Verimag, Verimag, Centre Équation, 38610 Gières (2009)
12. Corin, R., den Hartog, J.: A probabilistic Hoare-style logic for game-based cryptographic proofs. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP (2), vol. 4052 of Lecture Notes in Computer Science, pp. 252–263. Springer, Heidelberg (2006)
13. Damgard, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, pp. 445–456. Springer, London (1992)
14. Datta, A., Derek, A., Mitchell, J.C., Warinschi, B.: Computationally sound compositional logic for key exchange protocols. In: CSFW '06: Proceedings of the 19th IEEE Workshop on Computer Security Foundations, pp. 321–334. IEEE Computer Society, Washington (2006)
15. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **IT-22**, 644–654 (1976)
16. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. J. Cryptol. **1**(2), 77–94 (1988)
17. Halevi, S.: A plausible approach to computer-aided cryptographic proofs. http://theory.lcs.mit.edu/~shaih/pubs.html (2005)
18. Okamoto, T., Pointcheval, D.: React: Rapid enhanced-security asymmetric cryptosystem transform. In: CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology, pp. 159–175. Springer, London(2001)
19. Pointcheval, D.: Chosen-ciphertext security for any one-way cryptosystem. In: PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography, pp. 129–146. Springer, London (2000)
20. Rabin, M.O.: Digitalized signatures as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge (1979)
21. Shoup, V.: OAEP reconsidered. J. Cryptol. **15**(4), 223–249 (2002)
22. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs http://eprint.iacr.org/2004/332 (2004)

23. Soldera, D., Seberry, J., Qu, C.: The analysis of Zheng–Seberry scheme. In: Batten, L.M., Seberry, J. (eds.) ACISP, vol. 2384 of Lecture Notes in Computer Science, pp. 159–168. Springer, Heidelberg (2002)
24. Tarento, S.: Machine-checked security proofs of cryptographic signature schemes. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) Computer Security–ESORICS 2005, vol. 3679 of Lecture Notes in Computer Science, pp. 140–158. Springer, Heidelberg (2005)
25. Zheng, Y., Seberry, J.: Immunizing public key cryptosystems against chosen ciphertext attacks. IEEE J. Sel. Areas Commun. **11**(5), 715–724 (1993)

Towards Automated Proofs for Asymmetric Encryption Schemes in the Random Oracle Model

Judicaël Courant, Marion Daubignard, Cristian Ene,
Pascal Lafourcade, Yassine Lakhnech*
Université Grenoble 1, CNRS, VERIMAG
firstname.last@imag.fr

ABSTRACT

Chosen-ciphertext security is by now a standard security property for asymmetric encryption. Many generic constructions for building secure cryptosystems from primitives with lower level of security have been proposed. Providing security proofs has also become standard practice. There is, however, a lack of automated verification procedures that analyze such cryptosystems and provide security proofs. This paper presents an automated procedure for analyzing generic asymmetric encryption schemes in the random oracle model. This procedure has been applied to several examples of encryption schemes among which the construction of Bellare-Rogaway 1993, of Pointcheval at PKC'2000 and REACT.

Categories and Subject Descriptors: E.3 DATA ENCRYPTION: Public key cryptosystems

General Terms: Security, verification.

Keywords: Hoare logics, asymmetric encryption, provable security, automated proofs, random oracle model.

1. INTRODUCTION

Our day-to-day lives increasingly depend upon information and our ability to manipulate it securely. This requires solutions based on cryptographic systems (primitives and protocols). In 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, the first cryptosystem with a reductionist security proof appeared (Rabin 1979). The next breakthrough towards formal proofs of security was the adoption of computational security for the purpose of rigorously defining the security of cryptographic schemes. In this framework, a system is *provably secure* if there is a polynomial-time reduction proof from a hard problem to an attack against the security of the system. 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 are achieved. While research in the field of provable cryptography has achieved tremendous progress towards rigorously defining the functionalities and requirements of many cryptosystems, little has been done for developing computer-aided proof methods or more generally for investigating a proof theory for cryptosystems as it exists for imperative programs, concurrent systems, reactive systems, etc...

In this paper, we present an automated proof method for analyzing generic asymmetric encryption schemes in the random oracle model (ROM). Generic encryption schemes aim at transforming schemes with weak security properties, such as one-wayness, into schemes with stronger security properties, especially security against chosen ciphertext attacks. Examples of generic encryption schemes are [11, 23, 21, 5, 6, 19, 18, 17]. The paper contains two main contributions. The first one is a compositional Hoare logic for proving IND-CPA-security. That is, we introduce a simple programming language (to specify encryption algorithms that use one-way functions and hash functions) and an assertion language that allows to state invariants and axioms and rules to establish such invariants. Compositionality of the Hoare logic means that the reasoning follows the structure of the program that specifies the encryption oracle. The assertion language consists of three atomic predicates. The first predicate allows us to express that the value of a variable is indistinguishable from a random value even when given the values of a set of variables. The second predicate allows us to state that it is computationally infeasible to compute the value of a variable given the values of a set of variables. Finally, the third predicate allows us to state that the value of a variable has not been submitted to a hash function.

Transforming the Hoare logic into an (incomplete) automated verification procedure is quite standard. Indeed, we can interpret the logic as a set of rules that tell us how to propagate the invariants backwards. We have done this for our logic resulting in a verification procedure implemented in less than 250 lines of CAML. We have been able to automatically verify IND-CPA security of several schemes among which [5, 18, 17]. Our Hoare logic is incomplete for two main reasons. First, IND-CPA security is an observational equivalence-based property, while with our Hoare logic we establish invariants. Nevertheless, as shown in Proposition 3.1, we can use our Hoare logic to prove IND-CPA security at the price of completeness. That is, we prove a stronger property than IND-CPA. The second reason, which we think is less important, is that for efficiency

*This work is partially supported by the project AVOTE, SCALP and SFINCS

CCS'08, October 27–31, 2008, Alexandria, Virginia, USA.

reasons some axioms are stronger than needed.

The second contribution of the paper presents a simple criterion for plaintext awareness (PA). Plaintext awareness has been introduced by Bellare and Rogaway in [6]. It has then been refined in [4] such that if an encryption scheme is PA and IND-CPA then it is IND-CCA. Intuitively, PA ensures that an adversary cannot generate a valid cipher without knowing the plaintext, and hence, the decryption oracle is useless for him. The definition of PA is complex and proofs of PA are also often complex. In this paper, we present a simple syntactic criterion that implies plaintext awareness. Roughly speaking the criterion states that the cipher should contain as a sub-string the hash of a bitstring that contains as substrings the plaintext and the random seed. This criterion applies for many schemes such as [5, 17, 18] and easy to check. Although (or maybe because) the criterion is simple, the proof of its correctness is complex.

Putting together these two contributions, we get a proof method for IND-CCA security.

An important feature of our method is that it is not based on a global reasoning and global program transformation as it is the case for the game-based approach [7, 20]. Indeed, both approaches can be considered complementary as the Hoare logic-based one can be considered as aiming at characterizing, by means of predicates, the set of contexts in which the game transformations can be applied safely.

Related work.

We restrict our discussion to work providing computational proofs for cryptosystems. In particular, this excludes symbolic verification (including ours). We mentioned above the game-based approach [7, 20, 15]. In [8, 9] B. Blanchet and D. Pointcheval developed a dedicated tool, CryptoVerif, that supports security proofs within the game-based approach. CryptoVerif is based on observational equivalence. The equivalence relation induces rewriting rules applicable in contexts that satisfy some properties. Invariants provable in our Hoare logic can be considered as logical representations of these contexts. Moreover, as we work with invariants, that is we follow a state-based approach, we need to prove results that link our invariants to game-based properties such as indistinguishability (cf. Proposition 3.1 and 3.12). Our verification method is fully automated. It focusses on asymmetric encryption in the random oracle model, while CryptoVerif is potentially applicable to any cryptosystem.

G. Barthe and S. Tarento were among the first to provide machine-checked proofs of cryptographic schemes without relying on the perfect cryptography hypothesis. They formalized the Generic Model and the Random Oracle Model in the Coq proof assistant, and used this formalization to prove hardness of the discrete logarithm [1], security of signed ElGamal encryption against interactive attacks [3], and of Schnorr signatures against forgery attacks [22]. They are currently working on formalizing the game-based approach in Coq [2]. D. Nowak provides in [16] an implementation in Coq of the game-based approach. He illustrates his framework by a proof of the semantic security of the encryption scheme ElGamal and its hashed version. Another interesting work is the Hoare-style proof system proposed by R. Corin and J. Den Hartog for game-based cryptographic proofs [10]. The main difference between our logic and theirs is that our assertion language does not manipulate probabilities explicitly and is at a higher level of abstraction. On the other hand, their logic is more general. In [12], Datta et al. present a computationally sound compositional logic for key exchange protocols. There is, however, no proof assistance provided for this logic neither.

Outline: In Section 2, we introduce notations used for defining our programming language and generic asymmetric encryption schemes. In Section 3, we present our method for proving IND-CPA security. In Section 4 we introduce a criterion to prove plaintext awareness. In Section 5 we explain the automated verification procedure derived from our Hoare logic. Finally, in Section 6 we conclude.

2. DEFINITIONS

We are interested in analyzing generic schemes for asymmetric encryption assuming ideal hash functions. That is, we are working in the *random oracle model* [13, 5]. Using standard notations, we write $H \xleftarrow{r} \Omega$ to denote that H is randomly chosen from the set of functions with appropriate domain. By abuse of notation, for a list $\vec{H} = H_1, \cdots, H_n$ of hash functions, we write $\vec{H} \xleftarrow{r} \Omega$ instead of the sequence $H_1 \xleftarrow{r} \Omega, \ldots, H_n \xleftarrow{r} \Omega$. We fix a finite set $\mathcal{H} = \{H_1, \ldots, H_n\}$ of hash functions and also a finite set Π of trapdoor permutations and $\mathcal{O} = \Pi \cup \mathcal{H}$. We assume an arbitrary but fixed ordering on Π and $\mathcal{H}$; just to be able to switch between set-based and vector-based notation. A *distribution ensemble* is a countable sequence of distributions $\{X_\eta\}_{\eta\in\mathbb{N}}$. We only consider distribution ensembles that can be constructed in polynomial time by probabilistic algorithms that have oracle access to $\mathcal{O}$. Given two distribution ensembles $X = \{X_\eta\}_{\eta\in\mathbb{N}}$ and $X' = \{X'_\eta\}_{\eta\in\mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A}, \eta, X, X') = \mathsf{Pr}[x \xleftarrow{r} X_\eta : \mathcal{A}^{\mathcal{O}}(x) = 1] - \mathsf{Pr}[x \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathcal{O}}(x) = 1].$$

We insist, above, that for each hash function H, the probabilities are also taken over the set of maps with the appropriate type. Let $\mathsf{Adv}(\eta, X, X') = \sup_{\mathcal{A}}(\mathsf{Adv}(\mathcal{A}, \eta, X, X'))$, the maximal advantage taken over all probabilistic polynomial-time algorithms. Then, two distribution ensembles X and X' are called *indistinguishable* if $\mathsf{Adv}(\eta, X, X')$ is negligible as a function of η and denoted by $X \sim X'$. In other words, for any polynomial-time (in η) probabilistic algorithm $\mathcal{A}$, $\mathsf{Adv}(\mathcal{A}, \eta, X, X')$ is negligible as a function of η. We insist that all security notions we are going to use are in the ROM, where all algorithms, including adversaries, are equipped with oracle access to the hash functions.

2.1 A simple programming language for encryption and decryption oracles

We introduce a simple programming language without loops in which the encryption and decryption oracles are specified. The motivation for fixing a notation is obvious: it is mandatory for developing an automatic verification procedure. Let Var be an arbitrary finite non-empty set of variables. Then, our programming language is built according to the following BNF described in Table 1, where for a bitstring $bs = b_1 \ldots b_k$ (b_i are bits), $bs[n, m] = b_n \ldots b_m$[1], and

[1]Notice that $bs[n, m] = \epsilon$, when $m < n$ and $bs[n, m] = bs[n, k]$, when $m > k$

$\mathcal{N}$ is the name of the oracle, c its body and x and y are the input and output variable respectively. Note the command $y[n,m]$ is only used in the decryptions, it is why we do not have to consider it in our Hoare logic. With this language we can sample an uniform value to x, apply a way function f and its inverse f^{-1}, a hash function, the exclusive-or, the concatenation and substring function, and perform an "if-then-else" (used only in the decryption function).

EXAMPLE 2.1. *The following command encodes the encryption scheme proposed by Bellare and Rogaway in [5] (shortly $\mathcal{E}(in_e; out_e) = f(r)||in_e \oplus G(r)||H(in_e||r)$):*

$$\begin{array}{l}\mathcal{E}(in_e, out_e):\\ r \xleftarrow{r} \{0,1\}^{\eta_0};\ a := f(r);\ g := G(r);\\ b := in_e \oplus g;\ s := in_e||r;\ c := H(s);\\ u := a||b||c;\ out_e := u;\\ \text{where, } f \in \Pi \text{ and } G, H \in \mathcal{H}.\end{array}$$

Semantics: In addition to the variables in Var, we consider variables $\mathbb{T}_{H_1}, \ldots, \mathbb{T}_{H_n}$. Variable $\mathbb{T}_{H_i}$ records the queries to the hash function H_i and *can not be accessed by the adversary*. Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to $\mathbb{T}_{H_i}$. A *state* associates a value in $\{0,1\}^*$ to each variable in Var and a list of pairs of values to $\mathbb{T}_H$. For simplicity of the presentation, we assume that all variables range over large domains, whose cardinalities are exponential in the security parameter η. $u \xleftarrow{r} \mathcal{U}$ is the uniform sampling of a value u from the appropriate domain. Given a state S, $S(\mathbb{T}_H).\mathsf{dom}$, respectively $S(\mathbb{T}_H).\mathsf{res}$, denotes the list obtained by projecting each pair in $S(\mathbb{T}_H)$ to its first, respectively second, element.

A program takes as input a *configuration* $(S, \vec{H}, (f, f^{-1}))$ and yields a distribution on configurations. A configuration is composed of a state S, a vector of hash functions $(H_1, \ldots, H_n)$ and a pair (f, f^{-1}) of a trapdoor permutation and its inverse. Let Γ denote the set of configurations and $\textsc{Dist}(\Gamma)$ the set of distributions on configurations. The semantics is given in Table 2, where $\delta(x)$ denotes the Dirac measure, i.e. $\mathsf{Pr}(x) = 1$. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma)$ to $\textsc{Dist}(\Gamma)$. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

A notational convention: It is easy to prove that commands preserve the values of $\vec{H}$ and (f, f^{-1}). Therefore, we can, without ambiguity, write $S' \xleftarrow{r} [\![c]\!](S, \vec{H}, (f, f^{-1}))$ instead of $(S', \vec{H}, (f, f^{-1})) \xleftarrow{r} [\![c]\!](S, \vec{H}, (f, f^{-1}))$. According to our semantics, commands denote functions that transform distributions on configurations to distributions on configurations. However, only distributions that are constructible are of interest. Their set is denoted by $\textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$ and is defined as the set of distributions of the form:
$[(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \vec{H} \xleftarrow{r} \Omega; S \xleftarrow{r} A^{\vec{H}, f, f^{-1}}() : (S, \vec{H}, f, f^{-1})]$
where A is an algorithm accessing f, f^{-1} and $\vec{H}$ and which records its queries to hashing oracles into the $\mathbb{T}_H$'s in S.

2.2 Asymmetric Encryption

We study generic constructions that convert any trapdoor permutation into a public-key encryption scheme. More specifically, our aim is to provide an automatic verification method for generic encryption schemes. We also adapt IND-CPA and IND-CCA security notions to our setting.

DEFINITION 2.1. *A* generic encryption scheme *is defined by a triple $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ such that:*

- *$\mathbb{F}$ is a* trapdoor permutation generator *that on input η generates an η-bit string trapdoor permutation (f, f^{-1})*
- *$\mathcal{E}(in_e, out_e) : \mathsf{c}$ and $\mathcal{D}(in_d, out_d) : \mathsf{c}'$ are oracle declarations for encryption and decryption.*

DEFINITION 2.2. *Let GE be a generic encryption scheme defined by $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$. Let $A = (A_1, A_2)$ be an adversary and $X \in \textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$. For $\alpha \in \{cpa, cca\}$ and $\eta \in \mathbb{N}$, let*

$$\begin{array}{rcl}\mathsf{Adv}^{ind-\alpha}_{A,GE}(\eta, X) & = & 2 * \mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X;\\ & & (x_0, x_1, s) \xleftarrow{r} A_1^{\mathcal{O}_1}(f); b \xleftarrow{r} \{0,1\};\\ & & S' \xleftarrow{r} [\![\mathcal{E}(x_b, out_e)]\!](S, \vec{H}, (f, f^{-1})) :\\ & & A_2^{\mathcal{O}_2}(f, x_0, x_1, s, S'(out_e)) = b] - 1\end{array}$$

where if $\alpha = cpa$ then $\mathcal{O}_1 = \mathcal{O}_2 = \vec{H}$ and if $\alpha = cca$ then $\mathcal{O}_1 = \mathcal{O}_2 = \vec{H} \cup \{\mathcal{D}\}$.

We insist, above, that A_1 outputs x_0, x_1 such that $|x_0| = |x_1|$ and that in the case of CCA, A_2 does not ask its oracle $\mathcal{D}$ to decrypt $S'(y)$. We say that GE is IND-α secure if $\mathsf{Adv}^{ind-\alpha}_{A,GE}(\eta, X)$ is negligible for any constructible distribution ensemble X and polynomial-time adversary A.

3. IND-CPA SECURITY

In this section, we present an effective procedure to verify IND-CPA security. The procedure may fail to prove a secure encryption scheme but never declares correct an insecure one. Thus, we sacrifice completeness for soundness, a situation very frequent in verification[2]. We insist that our procedure does not fail for any of the numerous constructions we tried.

We are aiming at developing a procedure that allows us to prove properties, i.e. invariants, of the encryption oracle. More precisely, the procedure annotates each control point of the encryption command with a set of predicates that hold at that point for any execution except with negligible probability. Given an encryption oracle $\mathcal{E}(\mathrm{in}_e, \mathrm{out}_e) : \mathsf{c}$ we want to prove that at the final control point, we have an invariant that tells us that the value of out_e is indistinguishable from a random value. As we will show, this implies IND-CPA security.

A few words now concerning how we present the verification procedure. First, we present in the assertion language the invariant properties we are interested in. Then, we present a set of rules of the form $\{\varphi\}c\{\varphi'\}$ meaning that execution of command c in any distribution that satisfies φ leads to a distribution that satisfies φ'. Using Hoare logic terminology, this means that the triple $\{\varphi\}c\{\varphi'\}$ is valid.

From now on, we suppose that the adversary has access to the hash functions $\vec{H}$, and he is given the trapdoor permutation f, but not its inverse f^{-1}.

3.1 The Assertion Language

Our assertion language is defined by the following grammar, where ψ defines the set of atomic assertions:

$$\begin{array}{l}\psi ::= \mathsf{Indis}(\nu x; V_1; V_2) \mid \mathsf{WS}(x; V) \mid \mathsf{H}(H, e)\\ \varphi ::= \mathsf{true} \mid \psi \mid \varphi \wedge \varphi,\end{array}$$

[2]We conjecture that the IND-CPA verification problem of schemes described in our language is undecidable.

$$
\begin{array}{rrcl}
\text{Command} & \mathsf{c} & ::= & x \xleftarrow{r} \mathcal{U} \mid x := f(y) \mid x := f^{-1}(y) \mid x := H(y) \mid x := y[n,m] \\
& & & \mid x := y \oplus z \mid x := y||z \mid \mathsf{if}\ x = y\ \mathsf{then\ c1\ else\ c2\ fi} \mid c; c \\
\text{Oracle declaration} & \mathcal{O} & ::= & \mathcal{N}(x,y) : \mathsf{c}
\end{array}
$$

Table 1: Language grammar.

$$
\begin{array}{l}
[\![x \xleftarrow{r} \mathcal{U}]\!](S,\vec{H},(f,f^{-1})) = [u \xleftarrow{r} \mathcal{U} : (S\{x \mapsto u\},\vec{H},(f,f^{-1}))] \\
[\![x := f(y)]\!](S,\vec{H},(f,f^{-1})) = \delta(S\{x \mapsto f(S(y))\},\vec{H},(f,f^{-1})) \\
[\![x := f^{-1}(y)]\!](S,\vec{H},(f,f^{-1})) = \delta(S\{x \mapsto f^{-1}(S(y))\},\vec{H},(f,f^{-1})) \\
[\![x := y[n,m]]\!](S,\vec{H},(f,f^{-1})) = \delta(S\{x \mapsto S(y)[n,m]\},\vec{H},(f,f^{-1})) \\
[\![x := H(y)]\!](S,\vec{H},(f,f^{-1})) = \\
\quad \left\{ \begin{array}{ll}
\delta(S\{x \mapsto v\},\vec{H},(f,f^{-1})) & ;\ \text{if } (S(y),v) \in \mathbb{T}_H \\
\delta(S\{x \mapsto v, \mathbb{T}_H \mapsto S(\mathbb{T}_H)\cdot(S(y),v)\},\vec{H},(f,f^{-1})) & ; \\
& \text{if } (S(y),v) \notin \mathbb{T}_H \text{ and } v = \vec{H}(H)(S(y))
\end{array} \right. \\
[\![x := y \oplus z]\!](S,\vec{H},(f,f^{-1})) = \delta(S\{x \mapsto S(y) \oplus S(z)\},\vec{H},(f,f^{-1})) \\
[\![x := y||z]\!](S,\vec{H},(f,f^{-1})) = \delta(S\{x \mapsto S(y)||S(z)\},\vec{H},(f,f^{-1})) \\
[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!] \\
[\![\mathsf{if}\ x\ \mathsf{then}\ \mathsf{c}_1\ \mathsf{else}\ \mathsf{c}_2\ \mathsf{fi}]\!](S,\vec{H},(f,f^{-1})) = \left\{ \begin{array}{ll}
[\![\mathsf{c}_1]\!](S,\vec{H},(f,f^{-1})) & \text{if } S(x) = 1 \\
[\![\mathsf{c}_2]\!](S,\vec{H},(f,f^{-1})) & \text{otherwise}
\end{array} \right. \\
[\![\mathcal{N}(v,y)]\!](S,\vec{H},(f,f^{-1})) = [\![c]\!](S\{x \mapsto v\},\vec{H},(f,f^{-1})) \text{ where } c \text{ is the body of } \mathcal{N}.
\end{array}
$$

Table 2: The semantics of the programming language

where $V_1, V_2 \subseteq \mathsf{Var}$ and e is an expression, that is, a variable x or the concatenation of a polynomial number of variables.

Intuitively, $\mathsf{Indis}(\nu x; V_1; V_2)$ is satisfied by a distribution on configurations, if any adversary has negligible probability to distinguish whether he is given the value of x or a random value, even when he is additionally given the values of the variables in V_1 and the image by the one-way permutation of those in V_2. The assertion $\mathsf{WS}(x; V)$ is satisfied by a distribution, if any adversary has negligible probability to compute the value of x, even when he is given the values of the variables in V. Finally, $\mathsf{H}(H, e)$ is satisfied when the value of e has not been submitted to the hash oracle H.

Notations: We use $\mathsf{Indis}(\nu x; V)$ instead of $\mathsf{Indis}(\nu x; V; \emptyset)$ and $\mathsf{Indis}(\nu x)$ instead of $\mathsf{Indis}(\nu x; \mathsf{Var})$. We also write V, x instead of $V \cup \{x\}$ and even x, y instead of $\{x, y\}$.

Formally, the meaning of the assertion language is defined by a satisfaction relation $X \models \varphi$, which tells us when a distribution on configurations X satisfies the assertion φ. In order to define the satisfaction relation $X \models \varphi$, we need to generalize indistinguishability as follows. Let X be a family of distributions in $\textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$ and V_1 and V_2 be sets of variables in Var. By $D(X, V_1, V_2)$ we denote the following distribution family (on tuples of bit-strings):

$$
\begin{array}{l}
D(X, V_1, V_2)_\eta = \\
\quad [(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X : (S(V_1), f(S(V_2)), \vec{H}, f)]
\end{array}
$$

Here $S(V_1)$ is the point-wise application of S to the elements of V_1 and $f(S(V_2))$ is the point-wise application of f to the elements of $S(V_2)$. We say that X and X' are $V_1; V_2$-*indistinguishable*, denoted by $X \sim_{V_1;V_2} X'$, if $D(X, V_1, V_2) \sim D(X', V_1, V_2)$.

Example 3.1. *Let S_0 be any state and let H_1 be a hash function. Recall that we are working in the ROM. Consider the following distributions: $X_\eta = [\beta; S := S_0\{x \mapsto u, y \mapsto H_1(u)\} : (S, \vec{H}, (f, f^{-1}))]$ and $X'_\eta = [\beta; u' \xleftarrow{r} \{0,1\}^{p(\eta)}; S := S_0\{x \mapsto u, y \mapsto H_1(u')\} : (S, \vec{H}, (f, f^{-1}))]$, where $\beta = \vec{H} \xleftarrow{r} \Omega; (f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); u \xleftarrow{r} \{0,1\}^{p(\eta)}$, where p is a polynomial. Then, we have $X \sim_{\{y\};\{x\}} X'$ but we do not have $X \sim_{\{y,x\};\emptyset} X'$, because then the adversary can query the value of $H_1(x)$ and match it to that of y.*

The satisfaction relation $X \models \psi$ is defined as follows:

- $X \models \mathsf{true}$, $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \mathsf{Indis}(\nu x; V_1; V_2)$ iff $X \sim_{V_1;V_2} [u \xleftarrow{r} \mathcal{U}; (S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X : (S\{x \mapsto u\}, \vec{H}, (f, f^{-1}))]$
- $X \models \mathsf{WS}(x; V)$ iff $\mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X : A(S(V)) = S(x)]$ is negligible, for any adversary A.
- $X \models \mathsf{H}(H, e)$ iff $\mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X : S(e) \in S(\mathbb{T}_H).\mathsf{dom}]$ is negligible.

The relation between our Hoare triples and semantic security is established by the following proposition that states that if the value of out_e is indistinguishable from a random value then the scheme considered is IND-CPA.

Proposition 3.1. *Let $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ be a generic encryption scheme. It is IND-CPA secure if $\{\mathsf{true}\}\mathsf{c}\{\mathsf{Indis}(\nu out_e; out_e, in_e)\}$ is valid.*

If $\{\mathsf{true}\}\mathsf{c}\{\mathsf{Indis}(\nu \mathrm{out}_e; \mathrm{out}_e, \mathrm{in}_e)\}$ holds then the encryption scheme is secure with respect to randomness of ciphertext. It is standard that randomness of ciphertext implies IND-CPA security.

3.2 A Hoare Logic for IND-CPA security

In this section we present our Hoare logic for IND-CPA security. We begin with a set of preservation axioms that tell us when an invariant established at the control point before a command can be transferred to the next control point. Then, for each command, except $x := f^{-1}(y)$, $x := y[n, m]$ and conditional, we present a set of specific axioms that

allow us to establish new invariants. The commands that are not considered are usually not used in encryption but only in decryption procedures, and hence, are irrelevant for IND-CPA security.

3.2.1 Generic preservation rules:

We assume $z \neq x$ and c is either $x \xleftarrow{r} \mathcal{U}$ or $x := y||t$ or $x = y \oplus t$ or $x := f(y)$ or $x := H(y)$ or $x := t \oplus H(y)$.

LEMMA 3.2. *The following axioms are sound, when $x \notin V_1 \cup V_2$:*

- *(G1)* $\{\mathsf{Indis}(\nu z; V_1; V_2)\}\ c\ \{\mathsf{Indis}(\nu z; V_1; V_2)\}$
- *(G2)* $\{\mathsf{WS}(z; V_1)\}\ c\ \{\mathsf{WS}(z; V_1)\}$
- *(G3)* $\{\mathsf{H}(H', e[e'/x])\}\ x := e'\ \{\mathsf{H}(H', e)\}$, *provided $H' \neq H$ in case $e' \equiv H(y)$. Here, $e[e'/x]$ is the expression obtained from e by replacing x by e'.*

3.2.2 Random Assignment:

LEMMA 3.3. *The following axioms are sound:*

- *(R1)* $\{\mathsf{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{Indis}(\nu x)\}$
- *(R2)* $\{\mathsf{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{H}(H, e)\}$ *if e is x or is of the form $e_1||x||e_2$, $x||e_2$ or $e_1||x$.*

Moreover, the following preservation axioms, where we assume $x \neq y$ [3]*, are sound:*

- *(R3)* $\{\mathsf{Indis}(\nu y; V_1; V_2)\}x \xleftarrow{r} \mathcal{U}\{\mathsf{Indis}(\nu y; V_1, x; V_2)\}$
- *(R4)* $\{\mathsf{WS}(y; V)\}x \xleftarrow{r} \mathcal{U}\{\mathsf{WS}(y; V, x)\}$

Axiom (R1) is obvious. Axiom (R2) takes advantage of the fact that $\mathcal{U}$ is a large set, or more precisely that its cardinality is exponential in the security parameter, and that since e contains the fresh generated x the probability that it has already been submitted to H is small. Axioms (R3) and (R4) state that the value of x cannot help an adversary in distinguishing the value of y from a random value in (R3) or computing its value in (R4). This is the case because the value of x is randomly sampled.

Henceforth, we write $x \in \mathsf{var}(e)$ to state that e is x or is of the form $e_1||x||e_2$, $x||e_2$ or $e_1||x$.

3.2.3 Hash Function:

LEMMA 3.4. *The following basic axioms are sound, when $x \neq y$, and α is either a constant or a variable:*

- *(H1)*$\{\mathsf{WS}(y; V) \wedge \mathsf{H}(H, y)\}x := \alpha \oplus \mathsf{H}(y)$ $\{\mathsf{Indis}(\nu x; V, x)\}$
- *(H2)*$\{\mathsf{H}(H, y)\}\ x := H(y)\{\mathsf{H}(H', e)\}$, *if e is x or is of the form $e_1||x||e_2$, $x||e_2$ or $e_1||x$.*
- *(H3)*$\{\mathsf{Indis}(\nu y; V; V', y) \wedge \mathsf{H}(H, y)\}x := H(y)$ $\{\mathsf{Indis}(\nu x; V, x; V', y)\}$ *if $y \notin V$*

Axiom (H1) captures the main feature of the random oracle model, namely that the hash function is a random function. Hence, if an adversary cannot compute the value of y and this latter has not been hashed yet then he cannot distinguish $H(y)$ from a random value. Axiom (H2) is similar to axiom (R2). Axiom (H3) uses the fact that the value of y can not be queried to the hash oracle.

[3]By $x = y$ we mean syntactic equality.

LEMMA 3.5. *The following preservation axioms are sound provided that $x \neq y$ and $z \neq x$:*

- *(H4)* $\{\mathsf{WS}(y; V) \wedge \mathsf{WS}(z; V) \wedge \mathsf{H}(H, y)\}x := H(y)$ $\{\mathsf{WS}(z; V, x)\}$
- *(H5)* $\{\mathsf{H}(H, e) \wedge \mathsf{WS}(z; y)\}x := H(y)\{\mathsf{H}(H, e)\}$, *if $z \in \mathsf{var}(e) \wedge x \notin \mathsf{var}(e)$*
- *(H6)* $\{\mathsf{Indis}(\nu y; V_1; V_2, y) \wedge \mathsf{H}(H, y)\}x := H(y)$ $\{\mathsf{Indis}(\nu y; V_1, x; V_2, y)\}$, *if $y \notin V_1$*
- *(H7)* $\{\mathsf{Indis}(\nu z; V_1, z; V_2) \wedge \mathsf{WS}(y; V_1 \cup V_2, z) \wedge \mathsf{H}(H, y)\}$ $x := H(y)\{\mathsf{Indis}(\nu z; V_1, z, x; V_2)\}$

The idea behind (H4) is that to the adversary the value of x is seemingly random so that it can not help to compute z. Axiom (H5) states that the value of e not having been hashed yet reminds true as long as e contains a variable z whose value is not computable out of y. (H6) and (H7) give necessary conditions to the preservation of indistinguishability that is based on the seemingly randomness of a hash value.

3.2.4 One-way Function:

LEMMA 3.6. *The following axiom is sound, when $y \notin V \cup \{x\}$:*

- *(O1)* $\{\mathsf{Indis}(\nu y; V; y)\}\ x := f(y)\ \{\mathsf{WS}(y; V, x)\}$.

Axiom (O1) captures the one-wayness of f.

LEMMA 3.7. *The following axioms are sound when $z \neq x$:*

- *(O2)* $\{\mathsf{Indis}(\nu z; V_1, z; V_2, y)\}\ x := f(y)$ $\{\mathsf{Indis}(\nu z; V_1, z, x; V_2)\}$, *if $z \neq y$*
- *(O3)* $\{\mathsf{WS}(z; V) \wedge \mathsf{Indis}(\nu y; V, z; y)\}\ x := f(y)$ $\{\mathsf{WS}(z; V, x)\}$

For one-way permutations, we also have the following axiom:

- *(P1)*$\{\mathsf{Indis}(\nu y; V_1; V_2, y)\}\ x := f(y)$ $\{\mathsf{Indis}(\nu x; V_1, x; V_2)\}$, *if $y \notin V_1 \cup V_2$*

Axiom (O2) is obvious since $f(y)$ is given to the adversary in the precondition and axiom (O3) follows from the fact that y and z are independent. Axiom (P1) simply ensues from the fact that f is a permutation.

3.2.5 The Xor operator

In the following axioms, we assume $y \neq z$.

LEMMA 3.8. *The following axiom is sound when $y \notin V_1 \cup V_2$:*

- *(X1)* $\{\mathsf{Indis}(\nu y; V_1, y, z; V_2)\}x := y \oplus z$ $\{\mathsf{Indis}(\nu x; V_1, x, z; V_2)\}$,

Moreover, we have the following axioms that are sound provided that $t \neq x, y, z$.

- *(X2)* $\{\mathsf{Indis}(\nu t; V_1, y, z; V_2)\}x := y \oplus z$ $\{\mathsf{Indis}(\nu t; V_1, x, y, z; V_2)\}$
- *(X3)* $\{\mathsf{WS}(t; V, y, z)\}x := y \oplus z\{\mathsf{WS}(t; V, y, z, x)\}$

To understand axiom (X1) one should consider y as a key and think about x as the one-time pad encryption of z with the key y. Axioms (X2) and (X3) take advantage of the fact that is easy to compute x given y and z.

3.2.6 Concatenation:

LEMMA 3.9. *The following axioms are sound:*

- *(C1)* $\{\mathsf{WS}(y;V)\}\ x := y||z\ \{\mathsf{WS}(x;V)\}$, *if* $x \notin V$. *A dual axiom applies for* z.
- *(C2)* $\{\mathsf{Indis}(\nu y;V_1,y,z;V_2)\wedge\mathsf{Indis}(\nu z;V_1,y,z;V_2)\}\ x := y||z\ \{\mathsf{Indis}(\nu x;V_1;V_2)\}$, *if* $y,z \notin V_1 \cup V_2$
- *(C3)* $\{\mathsf{Indis}(\nu t;V_1,y,z;V_2)\}x := y||z$ $\{\mathsf{Indis}(\nu t;V_1,x,y,z;V_2)\}$, *if* $t \neq x,y,z$
- *(C4)* $\{\mathsf{WS}(t;V,y,z)\}\ x := y||z\ \{\mathsf{WS}(t;V,y,z,x)\}$, *if* $t \neq x,y,z$

(C1) states that if computing a substring of x out of the elements of V is hard, then so is computing x itself. The idea behind (C2) is that y and z being random implies randomness of x, with respect to V_1 and V_2. Eventually, x being easily computable from y and z accounts for rules (C3) and (C4).

In addition to the axioms above, we have the usual sequential composition and consequence rules of the Hoare logic. In order to apply the consequence rule, we use entailment (logic implication) between assertions as in Lemma 3.10.

LEMMA 3.10. *Let* $X \in \mathrm{DIST}(\Gamma,\vec{H},\mathbb{F})$ *be a distribution ensemble:*

1. *If* $X \models \mathsf{Indis}(\nu x;V_1;V_2)$, $V_1' \subseteq V_1$ *and* $V_2' \subseteq V_1 \cup V_2$ *then* $X \models \mathsf{Indis}(\nu x;V_1';V_2')$.
2. *If* $X \models \mathsf{WS}(x;V')$ *and* $V \subseteq V'$ *then* $X \models \mathsf{WS}(x;V)$.
3. *If* $X \models \mathsf{Indis}(\nu x;V_1;V_2 \cup \{x\})$ *and* $V \subseteq V_1 \setminus \{x\}$ *then* $X \models \mathsf{WS}(x;V)$.

The soundness of the Hoare Logic follows by induction from the soundness of each axiom and soundness of the Consequence and Sequential composition rules.

PROPOSITION 3.11. *The Hoare triples of Section 3.2 are valid.*

EXAMPLE 3.2. *We illustrate our proposition with Bellare & Rogaway's generic construction [5].*

$1)\ r \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r;\mathsf{Var}) \wedge \mathsf{H}(G,r) \wedge \mathsf{H}(H,in_e||r)$
$2)\ a := f(r)$
$\mathsf{Indis}(\nu a;\mathsf{Var}-r) \wedge \mathsf{WS}(r;\mathsf{Var}-r) \wedge \mathsf{H}(G,r)\wedge$
$\mathsf{H}(H,in_e||r)$
$3)\ g := G(r)$
$\mathsf{Indis}(\nu a;\mathsf{Var}-r) \wedge \mathsf{Indis}(\nu g;\mathsf{Var}-r)\wedge$
$\mathsf{WS}(r;\mathsf{Var}-r) \wedge \mathsf{H}(H,in_e||r)$
$4)\ b := in_e \oplus g$
$\mathsf{Indis}(\nu a;\mathsf{Var}-r) \wedge \mathsf{Indis}(\nu b;\mathsf{Var}-g-r)\wedge$
$\mathsf{WS}(r;\mathsf{Var}-r) \wedge \mathsf{H}(H,in_e||r)$
$5)\ s := in_e||r$
$\mathsf{Indis}(\nu a;\mathsf{Var}-r-s) \wedge \mathsf{Indis}(\nu b;\mathsf{Var}-g-r-s)\wedge$
$\mathsf{WS}(s;\mathsf{Var}-r-s) \wedge \mathsf{H}(H,s)$
$6)\ c := H(s)$
$\mathsf{Indis}(\nu a;\mathsf{Var}-r-s) \wedge \mathsf{Indis}(\nu b;\mathsf{Var}-r-g-s)\wedge$
$\mathsf{Indis}(\nu c;\mathsf{Var}-r-s)$
$7)\ out_e := a||b||c$
$\mathsf{Indis}(\nu out_e;\mathsf{Var}-a-b-c-r-g-s)$

1) $(R1)$, $(R2)$, *and* $(R2)$.
2) $(P1)$, $(O1)$, $(G3)$, *and* $(G3)$.
3) $(H7)$, $(H1)$, $(H4)$, *and* $(G3)$.
4) $(X2)$, $(X1)$, $(X3)$, *and* $(G3)$.
5) $(G1)$, $(G1)$, $(C1)$, *and* $(G3)$.
6) $(H7)$, $(H7)$, *and* $(H1)$.
7) $(C2)$ *twice.*

3.3 Extensions

In this section, we show how our Hoare logic, and hence our verification procedure, can be adapted to deal with on one hand injective partially trapdoor one-way functions and on the other hand OW-PCA (probabilistic) functions. The first extension is motivated by Pointcheval's construction in [18] and the second one by the Rapid Enhanced-security Asymmetric Cryptosystem Transform (REACT) [17]. For obvious reasons, we cannot recall the definitions of the security of these functions; we explain them informally.

The first observation we have to make is that Proposition 3.1 is too demanding in case f is not a permutation. Therefore, we introduce a new predicate $\mathsf{Indis}_f(\nu x;V_1;V_2)$ whose meaning is as follows:

$X \models \mathsf{Indis}_f(\nu x;V_1;V_2)$ if and only if $X \sim_{V_1;V_2} [u \xleftarrow{r} \mathcal{U};(S,\vec{H},(f,f^{-1})) \xleftarrow{r} X : (S\{x \mapsto f(u)\},\vec{H},(f,f^{-1}))]$.

Notice that, when f is a bijection, $\mathsf{Indis}_f(\nu x;V_1;V_2)$ is equivalent to $\mathsf{Indis}(\nu x;V_1;V_2)$ (f_i can be the identity function as in the last step of Example 3.3 and 3.4). Now, let out_e, the output of the encryption oracle, have the form $a_1||\cdots||a_n$ with $a_i = f_i(x_i)$. Then, we can prove the following:

PROPOSITION 3.12. *We consider GE a generic encryption scheme of the form* $(\mathbb{F},\mathcal{E}(in_e,out_e) : \mathsf{c},\mathcal{D}(in_d,out_d) : \mathsf{c}')$.

If $\{\mathsf{true}\}\mathsf{c}\{\bigwedge_{i=1}^{n} \mathsf{Indis}_{f_i}(\nu a_i;a_1,\ldots,a_n,in_e)\}$ *is valid then GE is IND-CPA.*

Now, we introduce a new axiom for $\mathsf{Indis}_f(\nu x;V_1;V_2)$ that replaces axiom (P1) in case the one-way function f is not a permutation:

$$(P1') \quad \{\mathsf{Indis}(\nu y;V_1;V_2,y)\}\ x := f(y)\ \{\mathsf{Indis}_f(\nu x;V_1,x;V_2)\} \text{ if } y \notin V_1 \cup V_2$$

Clearly all preservation rules can be generalized for Indis_f.

Injective partially trapdoor one-way functions: In contrast to the previous section, we do not assume f to be a permutation. On the other hand, we demand a stronger property than one-wayness. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function and let $f^{-1} : \mathcal{Z} \to \mathcal{X}$ be such that $\forall z \in \mathsf{dom}(f^{-1})\exists y \in \mathcal{Y},\ z = f(f^{-1}(z),y)$. Here f^{-1} is a partial function. The function f is said *partially one-way*, if for any given $z = f(x,y)$, it is computationally impossible to compute a corresponding x. In order to deal with the fact that f is now partially one-way, we add the following axioms, where we assume $x,y \notin V \cup \{z\}$ and where we identify f and $(x,y) \mapsto f(x||y)$:

$$(\mathrm{PO1}) \quad \{\mathsf{Indis}(\nu x;V,x,y) \wedge \mathsf{Indis}(\nu y;V,x,y)\}\ z := f(x||y)\ \{\ \mathsf{WS}(x;V,z) \wedge \mathsf{Indis}_f(\nu z;V,z)\ \}$$

The intuition behind the first part of (PO1) is that f guarantees one-way secrecy of the x-part of $x||y$. The second part follows the same idea that (P1').

EXAMPLE 3.3. *We verify Pointcheval's transformer [18].*

1) $r \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r)$
2) $s \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu s; \mathsf{Var}) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, in_e||s)$
3) $w := in_e||s$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{WS}(w; \mathsf{Var} - s - w) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, w)$
4) $h := H(w)$
$\mathsf{Indis}(\nu r; \mathsf{Var} - w - s) \wedge \mathsf{Indis}(\nu h; \mathsf{Var} - w - s) \wedge \mathsf{H}(G, r)$
5) $a := f(r||h)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - s - w - h)$
$\wedge \mathsf{WS}(r; \mathsf{Var} - r - s - w - h) \wedge \mathsf{H}(G, r)$
6) $b := w \oplus G(r)$
$\mathsf{Indis}_f(\nu a; a, in_e) \wedge \mathsf{Indis}(\nu b; a, b, in_e)$
7) $out_e := a||b$
$\mathsf{Indis}_f(\nu a; a, in_e) \wedge \mathsf{Indis}(\nu b; a, b, in_e)$

1) $(R1)$ *and* $(R2)$*; 2)* $(R3)$*,* $(R1)$*,* $(G3)$ *and* $(R2)$*; 3)* $(C3)$*,* $(C1)$*,* $(G3)$*, and* $(G3)$*; 4)* $(H7)$*,* $(H1)$*, and* $(G3)$*; 5) New rule* $(PO1)$ *and* $(G3)$*; 6) Extension of* $(G1)$ *to* Indis_f*, and* $(H1)$*; 7) Extension of* $(G1)$ *to* Indis_f*, and* $(G1)$*.*

To conclude, we use the fact that $\mathsf{Indis}_f(\nu a; a, in_e)$ *and* $\mathsf{Indis}(\nu b; a, b, in_e)$ *implies* $\mathsf{Indis}_f(\nu a; a, b, in_e)$

OW-PCA: Some constructions such as REACT are based on probabilistic one-way functions that are difficult to invert even when the adversary has access to a plaintext checking oracle (PC), which on input a pair (m, c), answers whether c encrypts m. In order to deal with OW-PCA functions, we need to strengthen the meaning of our predicates allowing the adversary to access to the additional plaintext checking oracle. For instance, the definition of $\mathsf{WS}(x; V)$ becomes: $X \models \mathsf{WS}(x; V)$ iff $\mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X : A^{PCA}(S(V)) = S(x)]$ is negligible, for any adversary A. Now, we have to revisit Lemma 3.10 and the axioms that introduce $\mathsf{WS}(x; V)$ in the postcondition. It is, however, easy to check that they are valid.

EXAMPLE 3.4. *REACT [17]*

1) $r \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r; \mathsf{Var})$
2) $R \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu R; \mathsf{Var}) \wedge \mathsf{H}(G, R) \wedge$
$\mathsf{H}(H, R||in_e||f(R||r)||in_e \oplus G(R))$
3) $a := f(R||r)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - R) \wedge \mathsf{WS}(R; \mathsf{Var} - r - R) \wedge$
$\mathsf{H}(G, R) \wedge \mathsf{H}(H, R||in_e||a||in_e \oplus G(R))$
4) $g := G(R)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - R) \wedge \mathsf{Indis}(\nu g; \mathsf{Var} - r - R) \wedge$
$\mathsf{WS}(R; \mathsf{Var} - r - R) \wedge \mathsf{H}(H, R||in_e||a||in_e \oplus g)$
5) $b := in_e \oplus g$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - R) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} - g - r - R) \wedge$
$\mathsf{WS}(R; \mathsf{Var} - r - R) \wedge \mathsf{H}(H, R||in_e||a||b)$
6) $w := R||in_e||a||b$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - w - R)$
$\wedge \mathsf{Indis}(\nu b; \mathsf{Var} - g - r - w - R)$
$\wedge \mathsf{WS}(w; \mathsf{Var} - r - w - R) \wedge \mathsf{H}(H, w)$
7) $c := H(w)$
$\mathsf{Indis}_f(\nu a; a, b, c, in_e) \wedge \mathsf{Indis}(\nu b; a, b, c, in_e)$
$\wedge \mathsf{Indis}(\nu c; a, b, c, in_e)$
8) $out_e := a||b||c;$
$\mathsf{Indis}_f(\nu a; a, b, c, in_e) \wedge \mathsf{Indis}(\nu b; a, b, c, in_e)$
$\wedge \mathsf{Indis}(\nu c; a, b, c, in_e)$

1) $(R1)$
2) $(R3)$*,* $(R1)$*,* $(R2)$ *and* $(R2)$
3) $(PO1)$*,* $(G3)$ *and* $(G3)$*.*
4) Extension of $(H7)$ *to* Indis_f*,* $(H1)$*,* $(H4)$*, and* $(G3)$*.*
5) Extension of $(X2)$ *to* Indis_f*,* $(X1)$*,* $(X3)$*, and* $(G3)$*.*
6) Extension of $(G1)$ *to* Indis_f*,* $(G1)$*,* $(C1)$*, and* $(G3)$*.*
7) Extension of $(H7)$ *to* Indis_f*,* $(H7)$*, and* $(H1)$*.*
8) Extension of $(G1)$ *to* Indis_f*,* $(G1)$ *and* $(G1)$*.*

4. PLAINTEXT AWARENESS

Bellare and Rogaway introduced *plaintext awareness (PA)* in [6][4]. The motivation is to decompose IND-CCA security of an encryption scheme into IND-CPA and PA security. Indeed, a public-key encryption scheme that satisfies IND-CPA (in the ROM) and the original definition of PA is IND-CCA1 (in the ROM). PA has been refined in [4] such that if an encryption scheme is PA and IND-CPA then it is IND-CCA. Intuitively, plaintext awareness means that the decryption oracle can be simulated by a *plaintext extractor* that does not have access to the inverse permutation f^{-1}. Now we introduce a simple analysis that allows us to automatically verify that an encryption scheme is PA in the strong sense [4]. Hence, combined with the results of the previous sections we obtain an analysis that allows to verify IND-CCA security.

We recall the definition of PA-security following the notations and conventions of [4]. Let $GE = (\mathbb{F}, \mathcal{E}(\mathrm{in}_e, \mathrm{out}_e) : \mathsf{c}, \mathcal{D}(\mathrm{in}_d, \mathrm{out}_d) : \mathsf{c}')$ be a generic encryption scheme. An adversary B for plaintext awareness is given the public permutation f, oracle access to the encryption algorithm $\mathcal{E}$ and to the ideal hash functions $\vec{H} = H_1, \cdots, H_n$. His goal is to output a cipher-text that cannot be correctly decrypted by the plaintext extractor. Hence, the success of plaintext extractor K against B in the distribution $X \in \textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$ is defined by:

$$\begin{array}{l}\mathsf{Succ}^{\mathsf{pa}}_{K,B,GE}(\eta, X) = \\ \quad \mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); \\ \quad S'' \xleftarrow{r} [\![\mathcal{D}(y, \mathrm{out}_d)]\!](S', \vec{H}, (f, f^{-1})) : \\ \quad y \in C \vee (y \notin C \wedge K(hH, C, y, f) = S''(\mathrm{out}_d))]\end{array}$$

Here by $(hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f)$ we mean the following: run B on input f with oracle access to H_i, $i = 1, \cdots, n$ and $\mathcal{E}$ (which calls f and H_i), recording B's interaction with the hash functions in hH and his interaction with $\mathcal{E}$ in C. Thus, hH is a list $(hH_1, \cdots, hH_n)$ of lists. Each list $hH_i = ((h_1, v_1), \cdots, (h_{q_i}, v_{q_i}))$ records all of B's H_i-oracle queries $h_1, \cdots, h_{q_i}$ and the corresponding answers $v_1, \cdots, v_{q_i}$. The modified state S' is due to calls of the hash functions either by B or the encryption oracle. The list C records the cipher-texts received in reply to $\mathcal{E}$-queries [5]. Finally, y is B's challenge to the plaintext extractor K. Please notice that K wins whenever B outputs a value $y \in C$.

DEFINITION 4.1. *An encryption scheme given by* $GE = (\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ *is* PA-secure*, if there*

[4]While in the original work by Bellare and Rogaway and in subsequent ones, plaintext awareness includes semantic security, we prefer to separate plaintext extraction and semantic security.

[5]This list was not included in the original definition by Bellare and Rogaway. Without it only IND-CCA1 can be proved but not IND-CCA.

is a polynomial-time probabilistic algorithm K such that for every distribution $X \in \text{Dist}(\Gamma, \vec{H}, \mathbb{F})$ and adversary B, we have $1 - \mathsf{Succ}^{pa}_{K,B,GE}(\eta, X)$ is a negligible function in η.

The rest of the section is organized as follows. We first introduce a semantic condition on $\mathcal{D}$ that implies the existence of a plaintext extractor. Then, we provide a syntactic criterion that implies the semantic criterion.

In the remainder of this section, we consider an encryption scheme GE that uses the hash functions $\vec{H} = H_1, \cdots, H_n$. We assume that c' has the following form
$\mathsf{c}_1; h := H_1(t);$
$\mathsf{if}\ \mathcal{V}(\vec{x}, h) = v\ \mathsf{then}\ \text{out}_d := m\ \mathsf{else}\ \text{out}_d := \text{"error"}\ \mathsf{fi},$
where $\vec{x}$ is a vector of variables (possibly empty) and $\mathcal{V}$ is a function (possibly the identity in which case we do not write it) such that for given $\vec{x}$ and v, $Pr[r \xleftarrow{r} \mathcal{U} : \mathcal{V}(\vec{x}, r) = v]$ is negligible. Furthermore, we require that the hash function H_1 is not called in c_1 and that the encryption algorithm c makes exactly one call to the oracle H_1. Consider, for instance, the scheme in [5], $f(r)||\text{in}_e \oplus G(r)||H(\text{in}_e||r)$. Here, t gets assigned the value $\text{in}_e||r$. We call the condition $\mathcal{V}(\vec{x}, h) = v$ (or equivalently $\mathcal{V}(\vec{x}, H_1(t)) = v$) the "*sanity check*".

It allows us to discriminate valid cipher-text from arbitrary bit-string. We also assume that decryption behaves correctly with respect to encryption: if y is generated using the algorithm of encryption, then the value of t as computed by the decryption oracle coincides with the value used as argument in the call to H_1 by the encryption algorithm.

Example 4.1. *Bellare and Rogaway [5]:*
$\mathcal{D}(in_d = a^*||b^*||v, out_d)$:
$r^* := f^{-1}(a^*); g^* := G(r^*); m^* := b^* \oplus g^*; t := m^*||r^*;$
$h := H(t);$
if $h = v$ *then* $out_d := m^*$ *else* $out_d :=$ "error" *fi*

A semantic criterion for PA Our semantic criterion for PA-security is composed of three conditions. We begin with an informal presentation of these conditions and how they will enable us to construct a plaintext extractor.

1. The first condition says that there is an algorithm that checks whether a given bit-string t^*, that has been submitted to H_1 by B, corresponds to the challenge y. That is, if the tester answers "yes" (1), then t^* matches with the value of t as computed by the decryption oracle and additionally satisfies the sanity check; and if it answers "no" (0), then t^* does not satisfy the sanity check.

2. The second condition states that it is easy to compute the plaintext from t^*.

3. The third condition states that for each value of t there is at most one corresponding ciphertext y.

Assume now that these conditions are satisfied. Then, we can construct a plaintext extractor K as follows. Using the algorithm of the first condition, that we call the tester, scan the list hH_1 to find a suitable t^*. If none is found, answer "error". Otherwise, apply the algorithm of the second condition on the value found for t^* to extract the plaintext. The third condition ensures that each value of t^* corresponds to at most one ciphertext, which is necessary to ensure that the extracted plaintext is the correct one. Let us now tackle the formal treatment of these ideas.

Definition 4.2. *We say that GE a generic encryption scheme satisfies the* PA-semantic criterion*, if there exist efficient algorithms $\mathcal{T}$ and Ext that satisfy the following conditions:*

1. *The* tester *$\mathcal{T}$ takes as input (hH, C, y, t^*, f) and returns a value in $\{0, 1\}$. We require that for any adversary B and any distribution $X \in \text{Dist}(\Gamma, \vec{H}, \mathbb{F})$,*

$$1- \mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); S'' \xleftarrow{r} [\![\mathcal{D}(y, out_d)]\!](S', \vec{H}, (f, f^{-1})); t^* \xleftarrow{r} hH_1.\mathsf{dom}; b \xleftarrow{r} \mathcal{T}(hH, C, y, t^*, f) : \big(b = 1 \Rightarrow H_1(t^*) = H_1(S''(t)) \wedge \mathcal{V}(S''(x), H_1(t^*)) = S''(v)\big) \wedge \big(b = 0 \Rightarrow \mathcal{V}(S''(x), H_1(t^*)) \neq S''(v)\big)]$$

is negligible.

2. *For Ext, we require that for any adversary B and any distribution $X \in \text{Dist}(\Gamma, \vec{H}, \mathbb{F})$,*

$$1- \Pr[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); S'' \xleftarrow{r} [\![\mathcal{D}(y, out_d)]\!](S', \vec{H}, (f, f^{-1})) : \mathsf{Ext}(hH, C, y, S''(t), f) = S''(out_d)]$$

is negligible.

3. *Finally, we require that for any adversary B and any distribution $X \in \text{Dist}(\Gamma, \vec{H}, \mathbb{F})$,*

$$\mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, y', S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); S_1 \xleftarrow{r} [\![\mathcal{D}(y, out_d)]\!](S', \vec{H}, (f, f^{-1})); S_2 \xleftarrow{r} [\![\mathcal{D}(y', out_d)]\!](S', \vec{H}, (f, f^{-1})) : y \neq y' \wedge S_1(t) = S_2(t) \wedge S_1(out_d) \neq \text{"error"} \wedge S_2(out_d) \neq \text{"error"}]$$

is negligible.

Of course there are generic encryption schemes for which the conditions above are satisfied under the assumption that $\mathcal{T}$ has access to an extra oracle such as a plaintext checking oracle (PC), or a ciphertext validity-checking oracle (CV), which on input c answers whether c is a valid ciphertext. In this case, the semantic security of the scheme has to be established under the assumption that f is OW-PCA, respectively OW-CVA. Furthermore, our definition of the PA-semantic criterion makes perfect sense for constructions that apply to IND-CPA schemes such as Fujisaki and Okamoto's converter [14]. In this case, f has to be considered as the IND-CPA encryption oracle.

Given a tester $\mathcal{T}$ and an algorithm Ext as in Definition 4.2, we construct a plaintext extractor as follows:

$K^{\mathcal{T}, \mathsf{Ext}}(hH, C, y, f)$:
Let $L = \{t^* \mid t^* \in \mathsf{dom}(hH_1), \mathcal{T}(hH, C, y, t^*, f) = 1\}$
if $L = \epsilon$ then return "error" else $t^* \xleftarrow{r} L$;
return $\mathsf{Ext}(hH, C, y, t^*, f)$

Theorem 4.1. *Let GE be a generic encryption scheme that satisfies the PA-semantic criterion. Then, GE is PA-secure.*

An easy syntactic check that implies the PA-semantic criterion is as follows.

Definition 4.3. *A generic encryption scheme GE satisfies the* PA-syntactic criterion*, if the sanity check has the form $\mathcal{V}(t, h) = v$, where $\mathcal{D}$ is such that h is assigned $H_1(t)$, t is assigned $in_e||r$, in_e is the plaintext and $\mathcal{E}(in_e; r)$ is the ciphertext (i.e., r is the random seed of $\mathcal{E}$).*

It is not difficult to see that if *GE* satisfies the PA-syntactic criterion then it also satisfies the PA-semantic one with a tester $\mathcal{T}$ as follows (Ext is obvious):

Look in hH_1 for a bit-string s such that $\mathcal{E}(x^*; r^*) = y$, where y is the challenge and $x^*||r^* = s$.

Here are some examples that satisfy the syntactic criterion (we use $\cdot^*$ to denote the values computed by the decryption oracle):

EXAMPLE 4.2.
- *Bellare and Rogaway [5]:* $\mathcal{E}(in_e; r) = a||b||c = f(r)||in_e \oplus G(r)||H(in_e||r)$. *The "sanity check" of the decryption algorithm is* $H(m^*||r^*) = c^*$.

- *OAEP+ [19]:* $\mathcal{E}(in_e; r) = f(a||b||c)$, *where* $a = in_e \oplus G(r)$, $b = H'(in_e||r)$, $c = H(s) \oplus r$ *and* $s = in_e \oplus G(r)||H'(in_e||r)$. *The "sanity check" of the decryption algorithm has the form* $H'(m^*||r^*) = b^*$.

- *Fujisaki and Okamoto [14]: if* $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ *is a public encryption scheme (that is CPA) then* $\mathcal{E}(in_e; r) = \mathcal{E}'((in_e||r); H(in_e||r))$. *The "sanity check" of the decryption algorithm is:*
  $\mathcal{E}'(m^*||r^*; H(m^*||r^*)) = in_d$.

The PA-semantic criterion applies to the following constructions but not the syntactic one:

EXAMPLE 4.3.

- *Pointcheval [18]:*
  $\mathcal{E}(in_e; r; s) = f(r||H(in_e||s))||((in_e||s) \oplus G(r))$, *where* f *is a partially trapdoor one-way injective function. The "sanity check" of the decryption oracle* $\mathcal{D}(a||b)$ *has the form* $f(r^*||H(m^*||s^*)) = a^*$. *The tester looks in* hG *and* hH *for* r^* *and* $m^*||s^*$ *such that* $\mathcal{E}(m^*; r^*; s^*) = y$.

- *REACT [17]: This construction applies to any trapdoor one-way function (possibly probabilistic). It is quite similar to the construction in [5]:* $\mathcal{E}(in_e; R; r) = a||b||c = f(R; r)||in_e \oplus G(r)||H(R||in_e||a||b)$, *where* $a = f(R; r)$ *and* $b = in_e \oplus G(R)$. *The "sanity check" of the decryption algorithm is* $H(R^*||m^*||a^*||b^*) = c$. *For this construction, one can provide a tester* $\mathcal{T}$ *that uses a PCA oracle to check whether* a *is the encryption of* R *by* f. *Hence, the PA security of the construction under the assumption of the OW-PCA security of* f. *The tester looks in* hH *for* $R^*||m^*||a^*||b^*$ *such that* $c^* = H(R^*||m^*||a^*||b^*)$ *and* $a^* = f(R^*)$, *which can be checked using the CPA-oracle.*

And now some examples of constructions that do not satisfy the PA-semantic criterion (and hence, not the syntactic one):

EXAMPLE 4.4.
- *Zheng-Seberry Scheme [23]:*

  $\mathcal{E}(x; r) = a||b = f(r)||(G(r) \oplus (x||H(x)))$. *The third condition of the PA-semantic criterion is not satisfied by this construction. Actually, there is an attack [21] on the IND-CCA security of this scheme that exploits this fact.*

- *OAEP [6]:* $\mathcal{E}(in_e; r) = a = f(in_e||0^k \oplus G(r)||r \oplus H(s))$, *where* $s = in_e||0^k \oplus G(r)$. *Here the third condition is not satisfied.*

5. AUTOMATION

We can now fully automate our verification procedure of IND-CCA for the encryption schemes we consider as follows:

1. Automatically establish invariants
2. Check the syntactic criterion for PA.

Point 2 can be done by a simple syntactic analyzer taking as input the decryption program, but has not been implemented yet.

Point 1 is more challenging. The idea is, for a given program, to compute invariants backwards, starting with the invariant $\mathsf{Indis}(\nu\mathrm{out}_e; \mathrm{out}_e, \mathrm{in}_e)$ at the end of the program.

As several rules can lead to a same postcondition, we in fact compute a set of sufficient conditions at all points of the program: for each set $\{\phi_1, \ldots, \phi_n\}$ and each instruction c, we can compute a set of assertions $\{\phi'_1, \ldots, \phi'_m\}$ such that

1. for $i = 1, \ldots, m$, there exists j such that $\{\phi'_i\}c\{\phi_j\}$ can be derived using the rules given section 3.2,
2. and for all j and all ϕ' such that $\{\phi'\}c\{\phi_j\}$, there exists i such that ϕ' entails ϕ'_i and that this entailment relation can be derived using lemma 3.10.

Of course, this verification is potentially exponential in the number of instructions of the encryption program as each postcondition may potentially have several preconditions. However this is mitigated as

- the considered encryption scheme are generally implemented in a few instructions (around 10)
- we implement a simplification procedure on the computed set of invariants: if ϕ_i entails ϕ_j (for $i \neq j$), then we can safely delete ϕ_i from the set of assertions $\{\phi_1, \ldots, \phi_n\}$. In other words, we keep only the minimal preconditions with respect to strength in our computed set of invariants (the usual Hoare logic corresponds to the degenerated case where this set has a minimum element, called the weakest precondition).

In practice, checking Bellare & Rogaway generic construction is instantaneous.

We implemented that procedure as an Objective Caml program, taking as input a representation of the encryption program. This program is only 230 lines long and is available on the web page of the authors.

6. CONCLUSION

In this paper we proposed an automatic method to prove IND-CCA security of generic encryption schemes in the random oracle model. IND-CPA is proved using a Hoare logic and plaintext awareness using a syntactic criterion. It does not seem difficult to adapt our Hoare logic to allow a security proof in the concrete framework of provable security. Another extension of our Hoare logic could concern OAEP. Here, we need to express that the value of a given variable is indistinguishable from a random value as long as a value r has not been submitted to a hash oracle G. This can be done by extending the predicate $\mathsf{Indis}(\nu x; V_1; V_2)$. The details are future work.

7. REFERENCES

[1] 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.

[2] Gilles Barthe, Bejamin Grégoire, Romain Janvier, and Santiago Zanella Béguelin. A framework for language-based cryptographic proofs. In *2nd Informal ACM SIGPLAN Workshop on Mechanizing Metatheory*, Oct 2007.

[3] 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.

[4] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 26–45, London, UK, 1998. Springer-Verlag.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, New York, USA, November 1993. ACM, ACM.

[6] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[7] 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/`.

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

[9] 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.

[10] 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.

[11] Ivan Damgard. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, London, UK, 1992. Springer-Verlag.

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

[13] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptol.*, 1(2):77–94, 1988.

[14] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC '99: Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, pages 53–68, London, UK, 1999. Springer-Verlag.

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

[16] David Nowak. A framework for game-based security proofs. In *ICICS*, pages 319–333, 2007.

[17] Tatsuaki Okamoto and David Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, pages 159–175, London, UK, 2001. Springer-Verlag.

[18] David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, pages 129–146, London, UK, 2000. Springer-Verlag.

[19] Victor Shoup. Oaep reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

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

[21] David Soldera, Jennifer Seberry, and Chengxin Qu. The analysis of zheng-seberry scheme. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2002.

[22] 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*, volume 3679 of *Lecture Notes in Computer Science*, pages 140–158. Springer, 2005.

[23] Yuliang Zheng and Jennifer Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.

Automated Security Proof for Symmetric Encryption Modes. *

Martin Gagné[2], Pascal Lafourcade[1], Yassine Lakhnech[1], and Reihaneh Safavi-Naini[2]

[1] Université Grenoble 1, CNRS,Verimag, FRANCE
[2] Department of Computer Science, University of Calgary, Canada

Abstract. We presents a compositional Hoare logic for proving semantic security of modes of operation for symmetric key block ciphers. We propose a simple programming language to specify encryption modes and an assertion language that allows to state invariants and axioms and rules to establish such invariants. The assertion language consists of few atomic predicates. We were able to use our method to verify semantic security of several encryption modes including Cipher Block Chaining (CBC), Cipher Feedback mode (CFB), Output Feedback (OFB), and Counter mode (CTR).

1 Introduction

A block cipher algorithm (e.g. AES, Blowfish, DES, Serpent and Twofish) is a symmetric key algorithm that takes a fixed size input message block and produces a fixed size output block. A mode of operation is a method of using a block cipher on an arbitrary length message. Important modes of operation are Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher FeedBack mode (CFB), Output FeedBack (OFB), and Counter mode (CTR). Modes of operations have different applications and provide different levels of security and efficiency. An important question when a mode of operation is used for encryption is the level of security that the mode provides, assuming the underlying block cipher is secure. The answer to this question is not straightforward. For example if one uses the naive ECB mode with a "secure" block cipher, then the encryption scheme obtained is not even IND-CPA secure. Others, like CBC or CTR, will provide confidentiality only if the initial vector (IV) is chosen adequately.

Recent years have seen an explosion of new modes of operation for block cipher (IACBC, IAPM [19], XCB [23], TMAC [18, 20], HCTR [5], HCH [7], EMU [15], EMU* [12], PEP [6], OMAC [16, 17], TET [13], CMC [14], GCM [24], EAX [4], XEX [25], TAE, TCH, TBC [22, 28] to name only a few). These new modes of operation often offer improved security guarantees, or additional security features. They also tend to be more complex than the traditional modes of operations, and arguments for proving their security can similarly become much

* This work was supported by ANR SeSur SCALP, SFINCS, AVOTE and iCORE.

more complicated – sometimes so complicated that flaws in the security proofs could go unnoticed for years.

Proofs generated by automated verification tools can provide us with an independent argument for the security of modes of operation, thereby increasing our confidence in the security of cryptographic protocols. While the rules used by the prover must also be proven by humans, and are therefore also susceptible to error, they tend to be much simpler than the protocols they will be used to check, which ensures that mistakes are far less likely to go unnoticed. In this paper, we take a first step towards building an automated prover for modes of operation, and show how to automatically generate proofs for many traditional block cipher modes of operation.

Contributions: We propose a compositional Hoare logic for proving semantic security of modes of operation for symmetric key block ciphers. We notice that many modes use a small set of operations such as xor, concatenation, and selection of random values. We introduce a simple programming language to specify encryption modes and an assertion language that allows to state invariants and axioms and rules to establish such invariants. The assertion language requires only four predicates: one that allows us to express that the value of a variable is indistinguishable from a random value when given the values of a set of variables, one that states that an expression has not been yet submitted to the block cipher, and two bookkeeping predicates that allow us to keep track of 'fresh' random values and counters. Transforming the Hoare logic into an (incomplete) automated verification procedure is quite standard. Indeed, we can interpret the logic as a set of rules that tell us how to propagate the invariants backwards. Using our method, an automated prover could verify semantic security of several encryption modes including CBC, CFB, CTR and OFB. Of course our system does not prove ECB mode, because ECB is not semantically secure.

Related Work: Security of symmetric encryption modes have been studied for a long time by the cryptographers. In [1] the authors presented different concrete security notions for symmetric encryption in a concrete security framework. For instance, they give a security analysis of CBC mode. In [2] a security analysis of the encryption mode CBC-MAC [21]. In [26] they propose a new encryption mode called OCB for efficient authenticated encryption and provide a security analysis of this new mode. Many other works present proofs of encryption modes.

Other works try to encode security of symmetric encryption modes as a non-interference property for programs with deterministic encryption. For example, [9] presents a computationally sound type system with exact security bounds for such programs. This type system has been applied to verify some symmetric encryption modes. The logic presented in this paper can be used to give a more structured soundness proof for the proposed type system. Moreover, we believe that our logic is more expressive and can be more easily adapted to more encryption modes.

A first important feature of our method is that it is not based on a global reasoning and global program transformation as it is the case for the game-based approach [3, 27].

In [8], the authors proposed an automatic method for proving semantic security for asymmetric generic encryption schemes. Our work continues that line of work. We extend the input language and axioms of the Hoare logic of [8] in order to capture symmetric encryption modes.

OUTLINE: In Section 2 we introduce the material for describing the encryption modes. In Section 3, we present our Hoare Logic for analyzing the semantic security of encryption modes described with the grammar given in the previous section. Finally before concluding in the last section, we apply our method to some examples in Section 4.

2 Definitions

2.1 Notation and Conventions

For simplicity, over this paper, we assume that all variables range over large domains, whose cardinality is exponential in the security parameter η. We also assume that all programs have length polynomial in η.

A block cipher is a function $\mathcal{E} : \{0,1\}^k \times \{0,1\}^\eta \to \{0,1\}^\eta$ such that for each $K \in \{0,1\}^k$, $\mathcal{E}(K,\cdot)$ is a permutation. It takes as input a k-bit key and an η-bit message block, and returns an η-bit string. We often denote by $\mathcal{E}(x)$ the application of the block cipher to the message block x. We omit the key used every time to simplify the notation, but it is understood that a key was selected at random at the beginning of the experiment and remains the same throughout.

For a mode of operation M, we denote by $\mathcal{E}_M$ the encryption function described by M using block cipher $\mathcal{E}$.

For a probability distribution $\mathcal{D}$, we denote by $x \xleftarrow{\$} \mathcal{D}$ the operation of sampling a value x according to distribution $\mathcal{D}$. If S is a finite set, we denote by $x \xleftarrow{\$} S$ the operation of sampling x uniformly at random among the values in S.

Given two distribution ensembles $X = \{X_\eta\}_{\eta\in\mathbb{N}}$ and $X' = \{X'_\eta\}_{\eta\in\mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A},\eta,X,X') = \mathsf{Pr}[x \xleftarrow{\$} X_\eta : \mathcal{A}(x) = 1] - \mathsf{Pr}[x \xleftarrow{\$} X'_\eta : \mathcal{A}(x) = 1].$$

Two distribution ensembles X and X' are called *indistinguishable*, denoted by $X \sim X'$, if $\mathsf{Adv}(\mathcal{A},\eta,X,X')$ is negligible as a function of η for every probabilistic polynomial-time algorithm $\mathcal{A}$.

2.2 Grammar

We introduce our language for defining a generic encryption mode. The commands are given by the grammar of Figure 1, where:

- $x \xleftarrow{\$} \mathcal{U}$ denotes uniform sampling of a value and assigning it to x.

- $x := \mathcal{E}(y)$ denotes application of the block cipher $\mathcal{E}$ to the value of y and assigning the result to x.
- Similarly for $x := \mathcal{E}^{-1}(y)$, where $\mathcal{E}^{-1}$ denotes the inverse function of $\mathcal{E}$.
- $x := y \oplus z$ denotes application of the exclusive-or operator to the values of y and z and assigning the result to x.
- $x := y||z$ represents the concatenation of the values of y and z.
- $x := y[n, m]$ assigns to x the bits at positions between n and m in the bit-string value of y. I.e., for a bit-string $bs = b_1 \dots b_k$, where the b_i's are bits, $bs[n, m]$ denotes the bits-string $b_n \dots b_m$[1]. Then, $x := y[n, m]$ assigns $bs[n, m]$ to x, where bs is the value of y. Here, n and m are polynomials in the security parameter η.
- $x := y + 1$ increments by one the value of y and assigns the result to x. The operation is carried modulo 2^η.
- $c_1; c_2$ is the sequential composition of c_1 and c_2.

$$\begin{aligned} \mathrm{c} ::= {} & x \xleftarrow{\$} \mathcal{U} \mid x := \mathcal{E}(y) \mid x := \mathcal{E}^{-1}(y) \mid x := y \oplus z \mid x := y \| z \mid x := y[n, m] \mid \\ & \mid x := y + 1 \mid c_1; c_2 \end{aligned}$$

Fig. 1. Language grammar

2.3 Generic Encryption Mode

We can now formally define a mode of encryption.

Definition 1 (Generic Encryption Mode). *A generic encryption mode M is represented by $\mathcal{E}_M(m_1|\dots|m_i, c_0|\dots|c_i)$: **var** $\boldsymbol{x_i}; \mathsf{c}_i$, where $\boldsymbol{x_i}$ is the set of variables used in c_i, all commands of c_i are built using the grammar described in Figure 1, each m_j is a message blocks, and each c_j is a cipher block, both of size n according to the input length of the block cipher $\mathcal{E}$.*

We add the additional block c_0 to the ciphertext because encryption modes are usually generate ciphertexts longer than the message. In all examples in this paper, c_0 will be the initialization vector (IV). The definition can easily be extended for encryption modes that also add one or more blocks at the end.

In Figure 2, we present the famous encryption mode $\mathcal{E}_{CBC}$ for a message of three blocks.

2.4 Semantics

In addition to the variables in Var,[2] we consider a variable $\mathcal{T}_E$ that records the values on which $\mathcal{E}$ was computed and cannot be accessed by the adversary.

[1] Notice that $bs[n, m] = \epsilon$, when $m < n$ and $bs[n, m] = bs[n, k]$, when $m > k$

[2] We denote by Var the complete set of variables in the program, whereas **var** denotes the set of variables in the program that are not input or output variables.

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$:
var z_1, z_2, z_3;
$IV \xleftarrow{\$} \mathcal{U}$;
$z_1 := IV \oplus m_1$;
$c_1 := \mathcal{E}(z_1)$;
$z_2 := c_1 \oplus m_2$;
$c_2 := \mathcal{E}(z_2)$;
$z_3 := c_2 \oplus m_3$;
$c_3 := \mathcal{E}(z_3)$;

Fig. 2. Description of $\mathcal{E}_{CBC}$

Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to $\mathcal{T}_E$. Given a state S, $S(\mathcal{T}_E).\mathsf{dom}$ and $S(\mathcal{T}_E).\mathsf{res}$ denote the lists obtained by projecting each pair in $S(\mathcal{T}_E)$ to its first and second element respectively.

The state also contains two sets of variables, F and C, which are used for bookkeeping purposes. The set F contains the variables with values that were sampled at random or obtained as a result of the computation of the block cipher, and have not yet been operated on. Those values are called *fresh* random values. The set C contains the variables whose value are the most recent increment of a counter that started at a fresh random value.

A program takes as input a *configuration* $(S, \mathcal{E})$ and yields a distribution on configurations. A configuration is composed of a state S, a block cipher $\mathcal{E}$. Let $\Gamma_{\mathcal{E}}$ denote the set of configurations and $\textsc{Dist}(\Gamma_{\mathcal{E}})$ the set of distributions on configurations. The semantics is given in Table 1. In the table, $\delta(x)$ denotes the Dirac measure, i.e. $\mathsf{Pr}[x] = 1$ and $\mathcal{T}_E \mapsto S(\mathcal{T}_E) \cdot (x, y)$ denotes the addition of element (x, y) to $\mathcal{T}_E$. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma_{\mathcal{E}})$ to $\textsc{Dist}(\Gamma_{\mathcal{E}})$. That is, let $\phi : \Gamma_{\mathcal{E}} \to \textsc{Dist}(\Gamma_{\mathcal{E}})$ be a function. Then, ϕ defines a unique function $\phi^* : \textsc{Dist}(\Gamma_{\mathcal{E}}) \to \textsc{Dist}(\Gamma_{\mathcal{E}})$ obtained by point-wise application of ϕ. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

A notational convention. It is easy to see that commands never alter $\mathcal{E}$. Therefore, we can, without ambiguity, write $S' \xleftarrow{\$} [\![c]\!](S, \mathcal{E})$ instead of $(S', \mathcal{E}) \xleftarrow{\$} [\![c]\!](S, \mathcal{E})$.

Here, we are only interested in distributions that can be constructed in polynomial time. We denote their set by $\textsc{Dist}(\mathit{\Gamma}, \mathcal{F})$, where $\mathcal{F}$ is a family of block ciphers, and is defined as the set of distributions of the form:

$$[\mathcal{E} \xleftarrow{\$} \mathcal{F}(1^\eta); S \xleftarrow{\$} [\![p]\!](I, \mathcal{E}) : (S, \mathcal{E})]$$

where p is a program with a polynomial number of commands, and I is the "initial" state, in which all variables are undefined and all lists and sets are empty.

$$
\begin{array}{l}
[\![x \xleftarrow{\$} \mathcal{U}]\!](S,\mathcal{E}) = [u \xleftarrow{\$} \mathcal{U} : (S\{x \mapsto u, \mathsf{F} \mapsto \mathsf{F} \cup \{x\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E})] \\
[\![x := \mathcal{E}(y)]\!](S,\mathcal{E}) = \\
\quad \begin{cases} \delta(S\{x \mapsto v, F \mapsto F \cup \{x\} \setminus \{y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \text{ if } (S(y), v) \in S(\mathcal{T}_E) \\ \delta(S\{x \mapsto v, F \mapsto F \cup \{x\} \setminus \{y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}, \mathcal{T}_E \mapsto S(\mathcal{T}_E) \cdot (S(y), v)\}, \mathcal{E}) \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } (S(y), v) \notin S(\mathcal{T}_E) \text{ and } v = \mathcal{E}(S(y)) \end{cases} \\
[\![x := \mathcal{E}^{-1}(y)]\!](S,\mathcal{E}) = \delta(S\{x \mapsto \mathcal{E}^{-1}(S(y)), F \mapsto F \setminus \{x,y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y \oplus z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y) \oplus S(z), F \mapsto F \setminus \{x,y,z\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y||z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)||S(z), F \mapsto F \setminus \{x,y,z\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y[n,m]]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)[n,m], F \mapsto F \setminus \{x,y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y+1]\!](S,\mathcal{E}) = \\
\quad \begin{cases} \delta(S\{x \mapsto S(y)+1, C \mapsto C \cup \{x\} \setminus \{y\}, F \mapsto F \setminus \{x,y\}\}, \mathcal{E}) \text{ if } y \in S(F) \text{ or } y \in S(C) \\ \delta(S\{x \mapsto S(y)+1, F \mapsto F \setminus \{x,y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \text{ otherwise} \end{cases} \\
[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]
\end{array}
$$

Table 1. The semantics of the programming language

2.5 Security Model

Ideal Cipher Model

We prove the modes of encryption secure in the ideal cipher model. That is, we assume that the block cipher is a pseudo-random function.[3] This is a standard assumption for proving the security of any block-cipher-based scheme.

The advantage of an algorithm $\mathcal{A}$ against a family of pseudo-random function is defined as follows.

Definition 2. *Let $P : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions and let $\mathcal{A}$ be an algorithm that takes an oracle and returns a bit. The* prf-advantage *of $\mathcal{A}$ is defined as follows.*

$$\mathsf{Adv}^{prf}_{\mathcal{A},P} = Pr[K \xleftarrow{\$} \{0,1\}^k; \mathcal{A}^{P(K,\cdot)} = 1] - Pr[R \xleftarrow{\$} \Phi_n; \mathcal{A}^{R(\cdot)} = 1]$$

where Φ_n is the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$.

The security of a symmetric mode of operation is usually proven by first showing that the mode of operation would be secure if $\mathcal{E}$ was a random function in Φ_n. As a result, an adversary $\mathcal{A}$ against the encryption scheme can be transformed into an adversary $\mathcal{B}$ against the block cipher (as a pseudo-random function) with a similar running time, such that $\mathcal{B}$'s prf-advantage is similar to $\mathcal{A}$'s advantage in breaking the encryption scheme.

Encryption Security

Semantic security for a mode of encryption is defined as follows.

[3] While block ciphers are really families of permutations, it is well known that pseudo-random permutations are indistinguishable from pseudo-random functions if the block size is large enough.

Definition 3. *Let* $\mathcal{E}_M(m_1|\dots|m_i, c_0|\dots|c_i) : \mathbf{var}\ \boldsymbol{x_i}; \mathsf{c}_i$ *be a generic encryption mode.* $A = (A_1, A_2)$ *be an adversary and* $X \in \textsc{Dist}(\Gamma, \mathcal{E})$*. For* $\eta \in \mathbb{N}$*, let*

$$\begin{aligned}\mathsf{Adv}^{ind-CPA}_{A,M}\ &(\eta, X)\\ &= 2 * \mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X;\\ &\quad (x_0, x_1, p, s) \xleftarrow{\$} A_1^{\mathcal{O}_1}(\eta); b \xleftarrow{\$} \{0,1\};\\ &\quad S' \xleftarrow{\$} [\![\mathsf{c}_p]\!](S\{m_1|\dots|m_p \mapsto x_b\}, \mathcal{E}) :\\ &\quad A_2^{\mathcal{O}_2}(x_0, x_1, s, S'(c_0|\dots|c_p)) = b] - 1\end{aligned}$$

where $\mathcal{O}_1 = \mathcal{O}_2$ *are oracles that take a pair* (m, j) *as input, where* m *is a string and* j *is the block length of* m*, and answers using the* j^{th} *algorithm in* $\mathcal{E}_M$*.* A_1 *outputs* x_0, x_1 *such that* $|x_0| = |x_1|$ *and are composed of* p *blocks. The mode of operation* M *is semantically (IND-CPA) secure if* $\mathsf{Adv}^{ind-CPA}_{A,M}(\eta, X)$ *is negligible for any constructible distribution ensemble* X *and polynomial-time adversary* A*.*

It is important to note that in this definition, an adversary against the scheme is only given oracle access to the encryption mode $\mathcal{E}_M$, and *not* to the block cipher $\mathcal{E}$ itself.

Our method verifies the security of an encryption scheme by proving that the ciphertext is indistinguishable from random bits. It is a classical result that this implies semantic security.

3 Proving Semantic Security

In this section, we present our Hoare logic for proving semantic (IND-CPA) security for generic encryption mode defined with our language. We prove that our logic is sound although not complete. Our logic can be used to annotate each command of our programming language with a set of invariants that hold at each point of the program for any execution.

3.1 Assertion Language

We consider new predicates in order to consider properties of symmetric encryption modes. We use a Hoare Logic based on the following invariants:

$$\begin{aligned}\varphi &::= \mathsf{true} \mid \varphi \wedge \varphi \mid \psi\\ \psi &::= \mathsf{Indis}(\nu x; V) \mid F(x) \mid \mathsf{E}(\mathcal{E}, e) \mid \mathit{Rcounter}(e),\end{aligned}$$

where $V \subseteq \mathsf{Var}$ and e is an expression constructible out of the variables used in the program and the grammar presented in Section 2. Intuitively:

$\mathsf{Indis}(\nu x; V)$: means that any adversary has negligible probability to distinguish whether he is given results of computations performed using the value of x or a random value, when he is given the values of the variables in V.

$\mathsf{E}(\mathcal{E}, e)$: means that the probability that the value $\mathcal{E}(e)$ has already been computed is negligible.

$F(e)$**:** means e is a fresh random value.
$RCounter(e)$**:** means that e is the most recent value of a counter that started at a fresh random value.

More formally, for each invariant ψ, we define that a distribution X satisfies ψ, denoted $X \models \psi$ as follows:

- $X \models \mathsf{true}$.
- $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \mathsf{Indis}(\nu x; V)$ iff $[(S, \mathcal{E}) \xleftarrow{\$} X : (S(x, V), \mathcal{E})] \sim [(S, \mathcal{E}) \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' = S\{x \mapsto u\} : (S'(x, V), \mathcal{E})]$
- $X \models \mathsf{E}(\mathcal{E}, e)$ iff $\mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X : S(e) \in S(\mathcal{T}_E).\mathsf{dom}]$ is negligible.
- $X \models F(e)$ iff $\mathsf{Pr}[(S, E) \xleftarrow{\$} X : e \in S(\mathsf{F})] = 1$.
- $X \models RCounter(e)$ iff $\mathsf{Pr}[(S, E) \xleftarrow{\$} X : e \in S(\mathsf{C})] = 1$.

3.2 Hoare Logic Rules

We present a set of rules of the form $\{\varphi\}\mathsf{c}\{\varphi'\}$, meaning that execution of command c in any distribution that satisfies φ leads to a distribution that satisfies φ'. Using Hoare logic terminology, this means that the triple $\{\varphi\}\mathsf{c}\{\varphi'\}$ is valid. We group rules together according to their corresponding commands. We do not provide rules for the commands $x := \mathcal{E}^{-1}(y)$ or $x := y[n, m]$ since those commands are only used during decryption.

Notation: For a set V, we write V, x as a shorthand for $V \cup \{x\}$, $V - x$ as a shorthand for $V \setminus \{x\}$, and $\mathsf{Indis}(\nu x)$ as a shorthand for $\mathsf{Indis}(\nu x; \mathsf{Var})$.

Random Assignment:

- (R1) $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{F(x) \wedge \mathsf{Indis}(\nu x) \wedge \mathsf{E}(\mathcal{E}, x)\}$
- (R2) $\{\mathsf{Indis}(\nu y; V)\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Indis}(\nu y; V, x)\}$

Increment:

- (I1) $\{F(y)\}\ x := y + 1\ \{RCounter(x) \wedge \mathsf{E}(\mathcal{E}, x)\}$
- (I2) $\{RCounter(y)\}\ x := y + 1\ \{RCounter(x) \wedge \mathsf{E}(E, x)\}$
- (I3) $\{\mathsf{Indis}(\nu z; V)\}\ x := y + 1\ \{\mathsf{Indis}(\nu z; V - x)\}$ if $z \neq x, y$ and $y \notin V$

Xor operator:

- (X1) $\{\mathsf{Indis}(\nu y; V, y, z)\}x := y \oplus z\{\mathsf{Indis}(\nu x; V, x, z)\}$ where $x, y, z \notin V$,
- (X2) $\{\mathsf{Indis}(\nu y; V, x)\}x := y \oplus z\{\mathsf{Indis}(\nu y; V)\}$ where $x \notin V$,
- (X3) $\{\mathsf{Indis}(\nu t; V, y, z)\}\ x := y \oplus z\ \{\mathsf{Indis}(\nu t; V, x, y, z)\}$ if $t \neq x, y, z$ and $x, y, z \notin V$
- (X4) $\{F(y)\}\ x := y \oplus z\ \{\mathsf{E}(\mathcal{E}, x)\}$ if $y \neq z$

Concatenation:

- (C1) $\{\mathsf{Indis}(\nu y; V, y, z)\} \wedge \{\mathsf{Indis}(\nu z; V, y, z)\}\ x := y \| z\ \{\mathsf{Indis}(\nu x; V, x)\}$ if $y, z \notin V$

– (C2) $\{\mathsf{Indis}(\nu t; V, y, z)\}\ x := y \| z\ \{\mathsf{Indis}(\nu t; V, x, y, z)\}$ if $t \neq x, y, z$

Block cipher:

– (B1) $\{\mathsf{E}(\mathcal{E}, y)\}\ x := \mathcal{E}(y)\ \{F(x) \wedge \mathsf{Indis}(\nu x) \wedge \mathsf{E}(\mathcal{E}, x)\}$
– (B2) $\{\mathsf{E}(\mathcal{E}, y) \wedge \mathsf{Indis}(\nu z; V)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu z; V)\}$ provided $z \neq x$
– (B3) $\{\mathsf{E}(\mathcal{E}, y) \wedge Rcounter(z)\}\ x := \mathcal{E}(z)\ \{Rcounter(z)\}$ provided $z \neq x$
– (B4) $\{\mathsf{E}(\mathcal{E}, y) \wedge \mathsf{E}(\mathcal{E}, z)\}\ x := \mathcal{E}(y)\ \{\mathsf{E}(\mathcal{E}, z)\}$ provided $z \neq x, y$
– (B5) $\{\mathsf{E}(\mathcal{E}, y) \wedge F(z)\}\ x := \mathcal{E}(y)\ \{F(z)\}$ provided $z \neq x, y$

Finally, we add a few rules whose purpose is to preserve invariants that are unaffected by the command.

Generic preservation rules:

Assume that $z \neq x, w, v$ and c is either $x \xleftarrow{\$} \mathcal{U}$, $x := w \| v$, $x := w \oplus v$, or $x := w + 1$:

– (G1) $\{\mathsf{Indis}(\nu z; V)\}\ \mathsf{c}\ \{\mathsf{Indis}(\nu z; V)\}$ provided $w, v \in V$
– (G2) $\{\mathsf{E}(\mathcal{E}, z)\}\ \mathsf{c}\ \{\mathsf{E}(\mathcal{E}, z)\}$
– (G3) $\{RCounter(z)\}\ \mathsf{c}\ \{RCounter(z)\}$
– (G4) $\{F(z)\}\ \mathsf{c}\ \{F(z)\}$

3.3 Proof Sketches

Due to space restrictions, we cannot present formal proofs of all our rules here. We present quick sketches instead to give the reader some intuition as to why each rule holds. The complete proofs are available in our full manuscript [11].

Rules for random assignment.

In rule (R1), $F(x)$ simply follows from the definition of $F(\cdot)$, and $\mathsf{Indis}(\nu x)$ should be obvious since x has just been sampled at random, independently of all other values. Also, since the block cipher has been computed only on a polynomial number of values, out of an exponential domain, the probability that x has been queried to the block cipher is clearly negligible. Rule (R2) is easily proven using the fact that, at this point, x is independent from all other values in the program.

Rules for increment.

For rules (I1) and (I2) the behavior of $RCounter(\cdot)$ easily follows from its definition. Note that since we have either $F(y)$ or $RCounter(y)$, y (and x) were obtained by repeatedly applying $+1$ to a random value r, i.e. $x = r + k$ for some number k. Since $\mathcal{E}$ was computed only on a polynomial number of values, the probability of being less than k away from one of those values is negligible, therefore the probability that x has been queried to the block cipher is negligible. In (I3), if $\mathsf{Indis}(\nu z; V)$ holds, then clearly $\mathsf{Indis}(\nu z; V - x)$ holds as well, and the values in $V - x$ are unchanged by the command.

Rules for Xor.

Rules (X1) and (X2) are proven by considering y as a one-time pad applied to z. As a result, one of x or y will be indistinguishable from random provided that the other is not known. For (X3), one simply notes that x is easy to construct from y and z, so if t is indistinguishable from random given y and z, then it is also indistinguishable from random given x, y and z. For rule (X4), since y is fresh, it is still independent from all other values, from z in particular. It then follows that x has the same distribution as y and is independent from all values except y and therefore, the probability that it has been queried to $\mathcal{E}$ is negligible for the same reason that y is.

Rules for concatenation.

Rules (C1) and (C2) follow simply from the observation that the concatenation of two independent random strings is a random string.

Rules for block cipher.

To prove (B1), in the Ideal Cipher Model, $\mathcal{E}$ is sampled at random among all possible functions $\{0,1\}^\eta \to \{0,1\}^\eta$. Since y has never been queried to the block cipher, $x := \mathcal{E}(y)$ is indistinguishable from an independent random value, and so possess the same invariants as if $x \xleftarrow{\$} \mathcal{U}$ had been executed. Rules (B2) to (B5) simply preserve invariants that are unaffected by the computation of the block cipher on a value that has never been queried before.

Generic preservation rules.

The conditions for applying those rules, particularly $z \neq x, w, v$ were designed specifically so that the command would have no effect on the invariant. The invariant is therefore preserved.

As a result of all this, we have the following:

Proposition 1. *In the Ideal Cipher Model, the Hoare triples given in the previous rules are valid.*

As a result, our method can be used to prove the semantic security of an encryption mode by proving that, from the adversary's point of view, the ciphertexts are indistinguishable from random bits.

Proposition 2. *Let $\mathcal{E}_M(m_1|\dots|m_i, c_0|\dots|c_i)$: **var** $\boldsymbol{x_i}; \mathsf{c}_i$ be a generic encryption mode describe with our language, and let $IO = \{m_1, \dots, m_i, c_0, \dots, c_i\}$. If $\{\mathsf{true}\}\mathsf{c}_i \bigwedge_{k=0}^{i} \{\mathsf{Indis}(\nu c_k; IO)\}$ is valid for every i, then $\mathcal{E}_M$ is IND-CPA secure in the Ideal Cipher Model.*

We conclude with the following, which states that our method of proving security of encryption modes is sound in the standard model.

Proposition 3. *Let $\mathcal{E}_M$ be an encryption mode proven secure in the Ideal Cipher Model using the method of Proposition 2. If there exists a standard model algorithm A such that $\mathsf{Adv}_{A,M}^{ind-CPA}(\eta, X)$ is non-negligible, then there exists an algorithm B such that $\mathsf{Adv}_{B,\mathcal{E}}^{prf}$ is non-negligible.*

4 Examples

In this section we apply our method to the traditional encryption modes (CBC), (CFB), (OFB) and (CTR) in respectively Figure 3, 4, 5 and 6. For simplicity, we consider messages consisting of only 3 blocks. The reader can easily be convinced that the same invariant propagation holds for any finite number of blocks. In order to prove IND-CPA security of these encryption schemes we have to prove that $c_0 = IV, c_1, c_2, c_3$ are indistinguishable from random bitstrings when given $m_1, m_2, m_3, c_0, c_1, c_2$ and c_3. Of course our method fails in analyzing ECB encryption mode and the "counter" version of CBC, which are two insecure operation modes.

CBC & CFB : In Figure 3 and 4, we describe the application of our set of rules on CBC and CFB examples. The analysis of these two encryption modes are similar.

OFB : The order of the commands in our description of OFB may seem strange, but it is not without reason. The variable z_{i+1} must be computed before c_i because no rule can preserve the invariant $\mathsf{E}(\mathcal{E}, z_i)$ through the computation of c_i.

CTR : This scheme is the only one of the four encryption modes we have studied that uses the increment command. The analysis is presented in Figure 6. We can see how the *RCounter* invariant is used for proving the IND-CPA security of this mode.

5 Conclusion

We proposed an automatic method for proving the semantic security of symmetric encryption modes. We introduced a small programming language in order to describe these modes. We construct a Hoare logic to make assertions about variables and propagate the assertions with the execution of the commands in the language. If the program which represents an encryption mode satisfies some invariants at the end of our automatic analysis then we conclude that the encryption mode is IND-CPA secure.

Future work: An obvious extension to our work would be to add a loop construct to our grammar. This would remove the necessity of having a different program for each message length within a mode of operation. We are also considering an extension of our work to prove CCA security of encryption modes using approaches such as the one proposed in [10] or the method proposed in [8]. Another more complex and challenging direction is to propose an extended version of our Hoare Logic in order to be able to analyze "modern" encryption modes which use more complex mathematical operation or primitives, or to try to use our method to prove security properties of other block-cipher based construction, such as unforgeability for block-cipher based MACs, or collision-resistance for block-cipher based hash functions.

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge F(IV) \wedge \mathsf{E}(\mathcal{E}, IV)\}$	(R1)
$z_1 := IV \oplus m_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, z_1)\}$	(X2)(X4)
$c_1 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}) \wedge F(c_1)\}$	(B1)
$z_2 := c_1 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2) \wedge \mathsf{E}(\mathcal{E}, z_2)\}$	(X2)(X4)
$c_2 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}) \wedge F(c_2)\}$	(B1)
$z_3 := c_2 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_3) \wedge \mathsf{E}(\mathcal{E}, z_3)\}$	(X2)(X4)
$c_3 := \mathcal{E}(z_3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_3) \wedge Indis(\nu c_3; \mathsf{Var})\}$	(B1)

Fig. 3. Analysis of CBC encryption mode

$\mathcal{E}_{CFB}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV) \wedge F(IV) \wedge E(\mathcal{E}, IV)\}$	(R1)
$z_1 := \mathcal{E}(IV)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu z_1) \wedge F(z_1)\}$	(B1)(B2)
$c_1 := z_1 \oplus m_1$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, c_1)\}$	(G1)(X1)(X4)
$z_2 := \mathcal{E}(c_1)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge F(z_2)\}$	(B1)(B2)
$c_2 := z_2 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{E}(\mathcal{E}, c_2)\}$	(X1) (X4)
$z_3 := \mathcal{E}(c_2)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge F(z_3)\}$	(B1)
$c_3 := z_3 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2)$	(X1)
	$\wedge\ \mathsf{Indis}(\nu c_3; \mathsf{Var} - z_3)\}$	

Fig. 4. Analysis of CFB encryption mode

$\mathcal{E}_{OFB}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge F(IV) \wedge \mathsf{E}(\mathcal{E}, IV)\}$	(R1)
$z_1 := \mathcal{E}(IV)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \{F(z_1) \wedge \mathsf{E}(\mathcal{E}, z_1) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var})\}$	(B1)(B2)
$z_2 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var}) \wedge \mathsf{E}(\mathcal{E}, z_2)$	(B1)(B2)
	$\wedge\ F(z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var})\}$	
$c_1 := m_1 \oplus z_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, z_2)$	(G1)(G2)(X1)
	$\wedge\ F(z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var})\}\}$	(G4)
$z_3 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, z_3)$	(B1)(B2)
	$\wedge\ \mathsf{Indis}(\nu z_2; \mathsf{Var}) \wedge F(z_3) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})\}$	(B2)
$c_2 := m_2 \oplus z_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})$	(X1)
$c_3 := m_3 \oplus z_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu c_3; \mathsf{Var} - z_3)$	(X1)

Fig. 5. Analysis of OFB encryption mode

$\mathcal{E}_{CTR}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge F(IV) \wedge \mathsf{E}(\mathcal{E}, IV)\}$	(R1)
$ctr1 := IV + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1)$	(I3)
	$\wedge\ Rcounter(ctr1) \wedge \mathsf{E}(\mathcal{E}, ctr1)\}$	(I1)
$z_1 := \mathcal{E}(ctr1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1) \wedge Rcounter(ctr1)$	(B2)(B3)
	$\wedge\ F(z_1) \wedge \mathsf{E}(\mathcal{E}, z_1) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var})\}$	(B1)
$c_1 := m_1 \oplus z_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1) \wedge Rcounter(ctr1)$	(G1)(G3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(X1)
$ctr2 := ctr1 + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(I3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ Rcounter(ctr2) \wedge \mathsf{E}(\mathcal{E}, ctr2)\}$	(I2)
$z_2 := \mathcal{E}(ctr2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge Rcounter(ctr2)$	(B1)
	$\wedge\ F(z_2) \wedge \mathsf{E}(\mathcal{E}, z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var})\}$	(B3)
$c_2 := m_2 \oplus z_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge Rcounter(ctr2)$	(G3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2)\}$	(X1)
$ctr3 := ctr2 + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(I3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, ctr3)$	(I2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge Rcounter(ctr3)\}$	(G1)
$z_3 := \mathcal{E}(ctr3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(B1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge Rcounter(ctr3)$	(B3)
	$\wedge\ F(z_3) \wedge \mathsf{E}(\mathcal{E}, z_3) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})\}$	
$c_3 := m_3 \oplus z_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(X1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2)$	
	$\wedge\ \mathsf{Indis}(\nu c_3; \mathsf{Var} - z_3)\}$	

Fig. 6. Analysis of CTR encryption mode

References

1. Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:394, 1997.
2. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.
3. 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/`.
4. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In *FSE*, pages 389–407, 2004.
5. Debrup Chakraborty and Mridul Nandi. An improved security bound for HCTR. pages 289–302, 2008.
6. Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2006.
7. Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-counter-hash approach. *IEEE Transactions on Information Theory*, 54(4):1683–1699, 2008.
8. Judicael Courant, Marion Daubignard, Cristian Ene, Pascal Lafourcade, and Yassine Lahknech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, (CCS'08)*, Alexandria, USA, October 2008.
9. Judicaël Courant, Cristian Ene, and Yassine Lakhnech. Computationally sound typing for non-interference: The case of deterministic encryption. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 364–375. Springer, 2007.
10. Anand Desai. New paradigms for constructing symmetric encryption schemes secure against chosen-ciphertext attack. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 394–412, London, UK, 2000. Springer-Verlag.
11. Martin Gagné, Pascal Lafourcade, Yassine Lakhnech, and Reihaneh Safavi-Naini. Automated security proof for symmetric encryption modes. Manuscript,2009. Available at `http://pages.cpsc.ucalgary.ca/~mgagne/TR_Asian.pdf`,.
12. Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2004.
13. Shai Halevi. Invertible universal hashing and the tet encryption mode. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007.
14. Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2003.
15. Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.

16. Tetsu Iwata and Kaoru Kurosawa. OMAC: One-key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.
17. Tetsu Iwata and Kaoru Kurosawa. On the security of a new variant of OMAC. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2003.
18. Tetsu Iwata and Kaoru Kurosawa. Stronger security bounds for OMAC, TMAC, and XCBC. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT*, volume 2904 of *Lecture Notes in Computer Science*, pages 402–415. Springer, 2003.
19. Charanjit S. Jutla. Encryption modes with almost free message integrity. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 529–544, London, UK, 2001. Springer-Verlag.
20. Kaoru Kurosawa and Tetsu Iwata. TMAC: Two-key CBC MAC. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2003.
21. Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit - a new construction. In *Fast Software Encryption Ç02, Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 2001.
22. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 31–46, London, UK, 2002. Springer-Verlag.
23. David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (XCB) mode of operation, 2007.
24. David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In *INDOCRYPT*, pages 343–355, 2004.
25. Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.
26. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 196–205, New York, NY, USA, 2001. ACM.
27. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. http://eprint.iacr.org/2004/332.
28. Peng Wang, Dengguo Feng, and Wenling Wu. On the security of tweakable modes of operation: TBC and TAE. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 274–287. Springer, 2005.

Automated Security Proof for Symmetric Encryption Modes. *

Martin Gagné[2], Pascal Lafourcade[1], Yassine Lakhnech[1], and Reihaneh Safavi-Naini[2]

[1] Université Grenoble 1, CNRS,Verimag, FRANCE
[2] Department of Computer Science, University of Calgary, Canada

Abstract. We presents a compositional Hoare logic for proving semantic security of modes of operation for symmetric key block ciphers. We propose a simple programming language to specify encryption modes and an assertion language that allows to state invariants and axioms and rules to establish such invariants. The assertion language consists of few atomic predicates. We were able to use our method to verify semantic security of several encryption modes including Cipher Block Chaining (CBC), Cipher Feedback mode (CFB), Output Feedback (OFB), and Counter mode (CTR).

1 Introduction

A block cipher algorithm (e.g. AES, Blowfish, DES, Serpent and Twofish) is a symmetric key algorithm that takes a fixed size input message block and produces a fixed size output block. A mode of operation is a method of using a block cipher on an arbitrary length message. Important modes of operation are Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher FeedBack mode (CFB), Output FeedBack (OFB), and Counter mode (CTR). Modes of operations have different applications and provide different levels of security and efficiency. An important question when a mode of operation is used for encryption is the level of security that the mode provides, assuming the underlying block cipher is secure. The answer to this question is not straightforward. For example if one uses the naive ECB mode with a "secure" block cipher, then the encryption scheme obtained is not even IND-CPA secure. Others, like CBC or CTR, will provide confidentiality only if the initial vector (IV) is chosen adequately.

Recent years have seen an explosion of new modes of operation for block cipher (IACBC, IAPM [19], XCB [23], TMAC [18, 20], HCTR [5], HCH [7], EMU [15], EMU* [12], PEP [6], OMAC [16, 17], TET [13], CMC [14], GCM [24], EAX [4], XEX [25], TAE, TCH, TBC [22, 28] to name only a few). These new modes of operation often offer improved security guarantees, or additional security features. They also tend to be more complex than the traditional modes of operations, and arguments for proving their security can similarly become much

* This work was supported by ANR SeSur SCALP, SFINCS, AVOTE and iCORE.

more complicated – sometimes so complicated that flaws in the security proofs could go unnoticed for years.

Proofs generated by automated verification tools can provide us with an independent argument for the security of modes of operation, thereby increasing our confidence in the security of cryptographic protocols. While the rules used by the prover must also be proven by humans, and are therefore also susceptible to error, they tend to be much simpler than the protocols they will be used to check, which ensures that mistakes are far less likely to go unnoticed. In this paper, we take a first step towards building an automated prover for modes of operation, and show how to automatically generate proofs for many traditional block cipher modes of operation.

Contributions: We propose a compositional Hoare logic for proving semantic security of modes of operation for symmetric key block ciphers. We notice that many modes use a small set of operations such as xor, concatenation, and selection of random values. We introduce a simple programming language to specify encryption modes and an assertion language that allows to state invariants and axioms and rules to establish such invariants. The assertion language requires only four predicates: one that allows us to express that the value of a variable is indistinguishable from a random value when given the values of a set of variables, one that states that an expression has not been yet submitted to the block cipher, and two bookkeeping predicates that allow us to keep track of 'fresh' random values and counters. Transforming the Hoare logic into an (incomplete) automated verification procedure is quite standard. Indeed, we can interpret the logic as a set of rules that tell us how to propagate the invariants backwards. Using our method, an automated prover could verify semantic security of several encryption modes including CBC, CFB, CTR and OFB. Of course our system does not prove ECB mode, because ECB is not semantically secure.

Related Work: Security of symmetric encryption modes have been studied for a long time by the cryptographers. In [1] the authors presented different concrete security notions for symmetric encryption in a concrete security framework. For instance, they give a security analysis of CBC mode. In [2] a security analysis of the encryption mode CBC-MAC [21]. In [26] they propose a new encryption mode called OCB for efficient authenticated encryption and provide a security analysis of this new mode. Many other works present proofs of encryption modes.

Other works try to encode security of symmetric encryption modes as a non-interference property for programs with deterministic encryption. For example, [9] presents a computationally sound type system with exact security bounds for such programs. This type system has been applied to verify some symmetric encryption modes. The logic presented in this paper can be used to give a more structured soundness proof for the proposed type system. Moreover, we believe that our logic is more expressive and can be more easily adapted to more encryption modes.

A first important feature of our method is that it is not based on a global reasoning and global program transformation as it is the case for the game-based approach [3, 27].

In [8], the authors proposed an automatic method for proving semantic security for asymmetric generic encryption schemes. Our work continues that line of work. We extend the input language and axioms of the Hoare logic of [8] in order to capture symmetric encryption modes.

OUTLINE: In Section 2 we introduce the material for describing the encryption modes. In Section 3, we present our Hoare Logic for analyzing the semantic security of encryption modes described with the grammar given in the previous section. Finally before concluding in the last section, we apply our method to some examples in Section 4.

2 Definitions

2.1 Notation and Conventions

For simplicity, over this paper, we assume that all variables range over large domains, whose cardinality is exponential in the security parameter η. We also assume that all programs have length polynomial in η.

A block cipher is a function $\mathcal{E} : \{0,1\}^k \times \{0,1\}^\eta \to \{0,1\}^\eta$ such that for each $K \in \{0,1\}^k$, $\mathcal{E}(K,\cdot)$ is a permutation. It takes as input a k-bit key and an η-bit message block, and returns an η-bit string. We often denote by $\mathcal{E}(x)$ the application of the block cipher to the message block x. We omit the key used every time to simplify the notation, but it is understood that a key was selected at random at the beginning of the experiment and remains the same throughout.

For a mode of operation M, we denote by $\mathcal{E}_M$ the encryption function described by M using block cipher $\mathcal{E}$.

For a probability distribution $\mathcal{D}$, we denote by $x \xleftarrow{\$} \mathcal{D}$ the operation of sampling a value x according to distribution $\mathcal{D}$. If S is a finite set, we denote by $x \xleftarrow{\$} S$ the operation of sampling x uniformly at random among the values in S.

Given two distribution ensembles $X = \{X_\eta\}_{\eta\in\mathbb{N}}$ and $X' = \{X'_\eta\}_{\eta\in\mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A},\eta,X,X') = \mathsf{Pr}[x \xleftarrow{\$} X_\eta : \mathcal{A}(x) = 1] - \mathsf{Pr}[x \xleftarrow{\$} X'_\eta : \mathcal{A}(x) = 1].$$

Two distribution ensembles X and X' are called *indistinguishable*, denoted by $X \sim X'$, if $\mathsf{Adv}(\mathcal{A},\eta,X,X')$ is negligible as a function of η for every probabilistic polynomial-time algorithm $\mathcal{A}$.

2.2 Grammar

We introduce our language for defining a generic encryption mode. The commands are given by the grammar of Figure 1, where:

- $x \xleftarrow{\$} \mathcal{U}$ denotes uniform sampling of a value and assigning it to x.

- $x := \mathcal{E}(y)$ denotes application of the block cipher $\mathcal{E}$ to the value of y and assigning the result to x.
- Similarly for $x := \mathcal{E}^{-1}(y)$, where $\mathcal{E}^{-1}$ denotes the inverse function of $\mathcal{E}$.
- $x := y \oplus z$ denotes application of the exclusive-or operator to the values of y and z and assigning the result to x.
- $x := y||z$ represents the concatenation of the values of y and z.
- $x := y[n, m]$ assigns to x the bits at positions between n and m in the bit-string value of y. I.e., for a bit-string $bs = b_1 \dots b_k$, where the b_i's are bits, $bs[n, m]$ denotes the bits-string $b_n \dots b_m$[1]. Then, $x := y[n, m]$ assigns $bs[n, m]$ to x, where bs is the value of y. Here, n and m are polynomials in the security parameter η.
- $x := y + 1$ increments by one the value of y and assigns the result to x. The operation is carried modulo 2^η.
- $c_1; c_2$ is the sequential composition of c_1 and c_2.

$$\begin{aligned} \mathrm{c} ::= {} & x \xleftarrow{\$} \mathcal{U} \mid x := \mathcal{E}(y) \mid x := \mathcal{E}^{-1}(y) \mid x := y \oplus z \mid x := y \| z \mid x := y[n, m] \mid \\ & \mid x := y + 1 \mid c_1; c_2 \end{aligned}$$

Fig. 1. Language grammar

2.3 Generic Encryption Mode

We can now formally define a mode of encryption.

Definition 1 (Generic Encryption Mode). *A generic encryption mode M is represented by $\mathcal{E}_M(m_1|\dots|m_i, c_0|\dots|c_i)$: **var** $\boldsymbol{x_i}; \mathsf{c}_i$, where $\boldsymbol{x_i}$ is the set of variables used in c_i, all commands of c_i are built using the grammar described in Figure 1, each m_j is a message blocks, and each c_j is a cipher block, both of size n according to the input length of the block cipher $\mathcal{E}$.*

We add the additional block c_0 to the ciphertext because encryption modes are usually generate ciphertexts longer than the message. In all examples in this paper, c_0 will be the initialization vector (IV). The definition can easily be extended for encryption modes that also add one or more blocks at the end.

In Figure 2, we present the famous encryption mode $\mathcal{E}_{CBC}$ for a message of three blocks.

2.4 Semantics

In addition to the variables in Var,[2] we consider a variable $\mathcal{T}_E$ that records the values on which $\mathcal{E}$ was computed and cannot be accessed by the adversary.

[1] Notice that $bs[n, m] = \epsilon$, when $m < n$ and $bs[n, m] = bs[n, k]$, when $m > k$

[2] We denote by Var the complete set of variables in the program, whereas **var** denotes the set of variables in the program that are not input or output variables.

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$:
var z_1, z_2, z_3;
$IV \xleftarrow{\$} \mathcal{U}$;
$z_1 := IV \oplus m_1$;
$c_1 := \mathcal{E}(z_1)$;
$z_2 := c_1 \oplus m_2$;
$c_2 := \mathcal{E}(z_2)$;
$z_3 := c_2 \oplus m_3$;
$c_3 := \mathcal{E}(z_3)$;

Fig. 2. Description of $\mathcal{E}_{CBC}$

Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to $\mathcal{T}_E$. Given a state S, $S(\mathcal{T}_E).\mathsf{dom}$ and $S(\mathcal{T}_E).\mathsf{res}$ denote the lists obtained by projecting each pair in $S(\mathcal{T}_E)$ to its first and second element respectively.

The state also contains two sets of variables, F and C, which are used for bookkeeping purposes. The set F contains the variables with values that were sampled at random or obtained as a result of the computation of the block cipher, and have not yet been operated on. Those values are called *fresh* random values. The set C contains the variables whose value are the most recent increment of a counter that started at a fresh random value.

A program takes as input a *configuration* $(S, \mathcal{E})$ and yields a distribution on configurations. A configuration is composed of a state S, a block cipher $\mathcal{E}$. Let $\Gamma_{\mathcal{E}}$ denote the set of configurations and $\textsc{Dist}(\Gamma_{\mathcal{E}})$ the set of distributions on configurations. The semantics is given in Table 1. In the table, $\delta(x)$ denotes the Dirac measure, i.e. $\mathsf{Pr}[x] = 1$ and $\mathcal{T}_E \mapsto S(\mathcal{T}_E) \cdot (x, y)$ denotes the addition of element (x, y) to $\mathcal{T}_E$. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma_{\mathcal{E}})$ to $\textsc{Dist}(\Gamma_{\mathcal{E}})$. That is, let $\phi : \Gamma_{\mathcal{E}} \to \textsc{Dist}(\Gamma_{\mathcal{E}})$ be a function. Then, ϕ defines a unique function $\phi^* : \textsc{Dist}(\Gamma_{\mathcal{E}}) \to \textsc{Dist}(\Gamma_{\mathcal{E}})$ obtained by point-wise application of ϕ. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

A notational convention. It is easy to see that commands never alter $\mathcal{E}$. Therefore, we can, without ambiguity, write $S' \xleftarrow{\$} [\![c]\!](S, \mathcal{E})$ instead of $(S', \mathcal{E}) \xleftarrow{\$} [\![c]\!](S, \mathcal{E})$.

Here, we are only interested in distributions that can be constructed in polynomial time. We denote their set by $\textsc{Dist}(\mathit{\Gamma}, \mathcal{F})$, where $\mathcal{F}$ is a family of block ciphers, and is defined as the set of distributions of the form:

$$[\mathcal{E} \xleftarrow{\$} \mathcal{F}(1^\eta); S \xleftarrow{\$} [\![p]\!](I, \mathcal{E}) : (S, \mathcal{E})]$$

where p is a program with a polynomial number of commands, and I is the "initial" state, in which all variables are undefined and all lists and sets are empty.

$$
\begin{array}{l}
[\![x \xleftarrow{\$} \mathcal{U}]\!](S,\mathcal{E}) = [u \xleftarrow{\$} \mathcal{U} : (S\{x \mapsto u, \mathsf{F} \mapsto \mathsf{F} \cup \{x\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E})] \\
[\![x := \mathcal{E}(y)]\!](S,\mathcal{E}) = \\
\quad \begin{cases} \delta(S\{x \mapsto v, F \mapsto F \cup \{x\} \setminus \{y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \text{ if } (S(y), v) \in S(\mathcal{T}_E) \\ \delta(S\{x \mapsto v, F \mapsto F \cup \{x\} \setminus \{y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}, \mathcal{T}_E \mapsto S(\mathcal{T}_E) \cdot (S(y), v)\}, \mathcal{E}) \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } (S(y), v) \notin S(\mathcal{T}_E) \text{ and } v = \mathcal{E}(S(y)) \end{cases} \\
[\![x := \mathcal{E}^{-1}(y)]\!](S,\mathcal{E}) = \delta(S\{x \mapsto \mathcal{E}^{-1}(S(y)), F \mapsto F \setminus \{x,y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y \oplus z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y) \oplus S(z), F \mapsto F \setminus \{x,y,z\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y||z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)||S(z), F \mapsto F \setminus \{x,y,z\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y[n,m]]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)[n,m], F \mapsto F \setminus \{x,y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \\
[\![x := y+1]\!](S,\mathcal{E}) = \\
\quad \begin{cases} \delta(S\{x \mapsto S(y)+1, C \mapsto C \cup \{x\} \setminus \{y\}, F \mapsto F \setminus \{x,y\}\}, \mathcal{E}) \text{ if } y \in S(F) \text{ or } y \in S(C) \\ \delta(S\{x \mapsto S(y)+1, F \mapsto F \setminus \{x,y\}, \mathsf{C} \mapsto \mathsf{C} \setminus \{x\}\}, \mathcal{E}) \text{ otherwise} \end{cases} \\
[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]
\end{array}
$$

Table 1. The semantics of the programming language

2.5 Security Model

Ideal Cipher Model

We prove the modes of encryption secure in the ideal cipher model. That is, we assume that the block cipher is a pseudo-random function.[3] This is a standard assumption for proving the security of any block-cipher-based scheme.

The advantage of an algorithm $\mathcal{A}$ against a family of pseudo-random function is defined as follows.

Definition 2. *Let $P : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions and let $\mathcal{A}$ be an algorithm that takes an oracle and returns a bit. The* prf-advantage *of $\mathcal{A}$ is defined as follows.*

$$\mathsf{Adv}^{prf}_{\mathcal{A},P} = Pr[K \xleftarrow{\$} \{0,1\}^k; \mathcal{A}^{P(K,\cdot)} = 1] - Pr[R \xleftarrow{\$} \Phi_n; \mathcal{A}^{R(\cdot)} = 1]$$

where Φ_n is the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$.

The security of a symmetric mode of operation is usually proven by first showing that the mode of operation would be secure if $\mathcal{E}$ was a random function in Φ_n. As a result, an adversary $\mathcal{A}$ against the encryption scheme can be transformed into an adversary $\mathcal{B}$ against the block cipher (as a pseudo-random function) with a similar running time, such that $\mathcal{B}$'s prf-advantage is similar to $\mathcal{A}$'s advantage in breaking the encryption scheme.

Encryption Security

Semantic security for a mode of encryption is defined as follows.

[3] While block ciphers are really families of permutations, it is well known that pseudo-random permutations are indistinguishable from pseudo-random functions if the block size is large enough.

Definition 3. *Let* $\mathcal{E}_M(m_1|\dots|m_i, c_0|\dots|c_i) : \mathbf{var}\ \boldsymbol{x_i}; \mathsf{c}_i$ *be a generic encryption mode.* $A = (A_1, A_2)$ *be an adversary and* $X \in \textsc{Dist}(\Gamma, \mathcal{E})$*. For* $\eta \in \mathbb{N}$*, let*

$$\begin{aligned}\mathsf{Adv}^{ind-CPA}_{A,M}\ &(\eta, X)\\ &= 2 * \mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X;\\ &\quad (x_0, x_1, p, s) \xleftarrow{\$} A_1^{\mathcal{O}_1}(\eta); b \xleftarrow{\$} \{0,1\};\\ &\quad S' \xleftarrow{\$} [\![\mathsf{c}_p]\!](S\{m_1|\dots|m_p \mapsto x_b\}, \mathcal{E}) :\\ &\quad A_2^{\mathcal{O}_2}(x_0, x_1, s, S'(c_0|\dots|c_p)) = b] - 1\end{aligned}$$

where $\mathcal{O}_1 = \mathcal{O}_2$ *are oracles that take a pair* (m, j) *as input, where* m *is a string and* j *is the block length of* m*, and answers using the* j^{th} *algorithm in* $\mathcal{E}_M$*.* A_1 *outputs* x_0, x_1 *such that* $|x_0| = |x_1|$ *and are composed of* p *blocks. The mode of operation* M *is semantically (IND-CPA) secure if* $\mathsf{Adv}^{ind-CPA}_{A,M}(\eta, X)$ *is negligible for any constructible distribution ensemble* X *and polynomial-time adversary* A*.*

It is important to note that in this definition, an adversary against the scheme is only given oracle access to the encryption mode $\mathcal{E}_M$, and *not* to the block cipher $\mathcal{E}$ itself.

Our method verifies the security of an encryption scheme by proving that the ciphertext is indistinguishable from random bits. It is a classical result that this implies semantic security.

3 Proving Semantic Security

In this section, we present our Hoare logic for proving semantic (IND-CPA) security for generic encryption mode defined with our language. We prove that our logic is sound although not complete. Our logic can be used to annotate each command of our programming language with a set of invariants that hold at each point of the program for any execution.

3.1 Assertion Language

We consider new predicates in order to consider properties of symmetric encryption modes. We use a Hoare Logic based on the following invariants:

$$\begin{aligned}\varphi &::= \mathsf{true} \mid \varphi \wedge \varphi \mid \psi\\ \psi &::= \mathsf{Indis}(\nu x; V) \mid F(x) \mid \mathsf{E}(\mathcal{E}, e) \mid \mathit{Rcounter}(e),\end{aligned}$$

where $V \subseteq \mathsf{Var}$ and e is an expression constructible out of the variables used in the program and the grammar presented in Section 2. Intuitively:

$\mathsf{Indis}(\nu x; V)$: means that any adversary has negligible probability to distinguish whether he is given results of computations performed using the value of x or a random value, when he is given the values of the variables in V.

$\mathsf{E}(\mathcal{E}, e)$: means that the probability that the value $\mathcal{E}(e)$ has already been computed is negligible.

$F(e)$**:** means e is a fresh random value.
$RCounter(e)$**:** means that e is the most recent value of a counter that started at a fresh random value.

More formally, for each invariant ψ, we define that a distribution X satisfies ψ, denoted $X \models \psi$ as follows:

- $X \models \mathsf{true}$.
- $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \mathsf{Indis}(\nu x; V)$ iff $[(S, \mathcal{E}) \xleftarrow{\$} X : (S(x, V), \mathcal{E})] \sim [(S, \mathcal{E}) \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' = S\{x \mapsto u\} : (S'(x, V), \mathcal{E})]$
- $X \models \mathsf{E}(\mathcal{E}, e)$ iff $\mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X : S(e) \in S(\mathcal{T}_E).\mathsf{dom}]$ is negligible.
- $X \models F(e)$ iff $\mathsf{Pr}[(S, E) \xleftarrow{\$} X : e \in S(\mathsf{F})] = 1$.
- $X \models RCounter(e)$ iff $\mathsf{Pr}[(S, E) \xleftarrow{\$} X : e \in S(\mathsf{C})] = 1$.

3.2 Hoare Logic Rules

We present a set of rules of the form $\{\varphi\}\mathsf{c}\{\varphi'\}$, meaning that execution of command c in any distribution that satisfies φ leads to a distribution that satisfies φ'. Using Hoare logic terminology, this means that the triple $\{\varphi\}\mathsf{c}\{\varphi'\}$ is valid. We group rules together according to their corresponding commands. We do not provide rules for the commands $x := \mathcal{E}^{-1}(y)$ or $x := y[n, m]$ since those commands are only used during decryption.

Notation: For a set V, we write V, x as a shorthand for $V \cup \{x\}$, $V - x$ as a shorthand for $V \setminus \{x\}$, and $\mathsf{Indis}(\nu x)$ as a shorthand for $\mathsf{Indis}(\nu x; \mathsf{Var})$.

Random Assignment:

- (R1) $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{F(x) \wedge \mathsf{Indis}(\nu x) \wedge \mathsf{E}(\mathcal{E}, x)\}$
- (R2) $\{\mathsf{Indis}(\nu y; V)\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Indis}(\nu y; V, x)\}$

Increment:

- (I1) $\{F(y)\}\ x := y + 1\ \{RCounter(x) \wedge \mathsf{E}(\mathcal{E}, x)\}$
- (I2) $\{RCounter(y)\}\ x := y + 1\ \{RCounter(x) \wedge \mathsf{E}(E, x)\}$
- (I3) $\{\mathsf{Indis}(\nu z; V)\}\ x := y + 1\ \{\mathsf{Indis}(\nu z; V - x)\}$ if $z \neq x, y$ and $y \notin V$

Xor operator:

- (X1) $\{\mathsf{Indis}(\nu y; V, y, z)\}x := y \oplus z\{\mathsf{Indis}(\nu x; V, x, z)\}$ where $x, y, z \notin V$,
- (X2) $\{\mathsf{Indis}(\nu y; V, x)\}x := y \oplus z\{\mathsf{Indis}(\nu y; V)\}$ where $x \notin V$,
- (X3) $\{\mathsf{Indis}(\nu t; V, y, z)\}\ x := y \oplus z\ \{\mathsf{Indis}(\nu t; V, x, y, z)\}$ if $t \neq x, y, z$ and $x, y, z \notin V$
- (X4) $\{F(y)\}\ x := y \oplus z\ \{\mathsf{E}(\mathcal{E}, x)\}$ if $y \neq z$

Concatenation:

- (C1) $\{\mathsf{Indis}(\nu y; V, y, z)\} \wedge \{\mathsf{Indis}(\nu z; V, y, z)\}\ x := y \| z\ \{\mathsf{Indis}(\nu x; V, x)\}$ if $y, z \notin V$

– (C2) $\{\mathsf{Indis}(\nu t; V, y, z)\}\ x := y \| z\ \{\mathsf{Indis}(\nu t; V, x, y, z)\}$ if $t \neq x, y, z$

Block cipher:

– (B1) $\{\mathsf{E}(\mathcal{E}, y)\}\ x := \mathcal{E}(y)\ \{F(x) \wedge \mathsf{Indis}(\nu x) \wedge \mathsf{E}(\mathcal{E}, x)\}$
– (B2) $\{\mathsf{E}(\mathcal{E}, y) \wedge \mathsf{Indis}(\nu z; V)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu z; V)\}$ provided $z \neq x$
– (B3) $\{\mathsf{E}(\mathcal{E}, y) \wedge Rcounter(z)\}\ x := \mathcal{E}(z)\ \{Rcounter(z)\}$ provided $z \neq x$
– (B4) $\{\mathsf{E}(\mathcal{E}, y) \wedge \mathsf{E}(\mathcal{E}, z)\}\ x := \mathcal{E}(y)\ \{\mathsf{E}(\mathcal{E}, z)\}$ provided $z \neq x, y$
– (B5) $\{\mathsf{E}(\mathcal{E}, y) \wedge F(z)\}\ x := \mathcal{E}(y)\ \{F(z)\}$ provided $z \neq x, y$

Finally, we add a few rules whose purpose is to preserve invariants that are unaffected by the command.

Generic preservation rules:

Assume that $z \neq x, w, v$ and c is either $x \xleftarrow{\$} \mathcal{U}$, $x := w \| v$, $x := w \oplus v$, or $x := w + 1$:

– (G1) $\{\mathsf{Indis}(\nu z; V)\}\ \mathsf{c}\ \{\mathsf{Indis}(\nu z; V)\}$ provided $w, v \in V$
– (G2) $\{\mathsf{E}(\mathcal{E}, z)\}\ \mathsf{c}\ \{\mathsf{E}(\mathcal{E}, z)\}$
– (G3) $\{RCounter(z)\}\ \mathsf{c}\ \{RCounter(z)\}$
– (G4) $\{F(z)\}\ \mathsf{c}\ \{F(z)\}$

3.3 Proof Sketches

Due to space restrictions, we cannot present formal proofs of all our rules here. We present quick sketches instead to give the reader some intuition as to why each rule holds. The complete proofs are available in our full manuscript [11].

Rules for random assignment.

In rule (R1), $F(x)$ simply follows from the definition of $F(\cdot)$, and $\mathsf{Indis}(\nu x)$ should be obvious since x has just been sampled at random, independently of all other values. Also, since the block cipher has been computed only on a polynomial number of values, out of an exponential domain, the probability that x has been queried to the block cipher is clearly negligible. Rule (R2) is easily proven using the fact that, at this point, x is independent from all other values in the program.

Rules for increment.

For rules (I1) and (I2) the behavior of $RCounter(\cdot)$ easily follows from its definition. Note that since we have either $F(y)$ or $RCounter(y)$, y (and x) were obtained by repeatedly applying $+1$ to a random value r, i.e. $x = r + k$ for some number k. Since $\mathcal{E}$ was computed only on a polynomial number of values, the probability of being less than k away from one of those values is negligible, therefore the probability that x has been queried to the block cipher is negligible. In (I3), if $\mathsf{Indis}(\nu z; V)$ holds, then clearly $\mathsf{Indis}(\nu z; V - x)$ holds as well, and the values in $V - x$ are unchanged by the command.

Rules for Xor.

Rules (X1) and (X2) are proven by considering y as a one-time pad applied to z. As a result, one of x or y will be indistinguishable from random provided that the other is not known. For (X3), one simply notes that x is easy to construct from y and z, so if t is indistinguishable from random given y and z, then it is also indistinguishable from random given x, y and z. For rule (X4), since y is fresh, it is still independent from all other values, from z in particular. It then follows that x has the same distribution as y and is independent from all values except y and therefore, the probability that it has been queried to $\mathcal{E}$ is negligible for the same reason that y is.

Rules for concatenation.

Rules (C1) and (C2) follow simply from the observation that the concatenation of two independent random strings is a random string.

Rules for block cipher.

To prove (B1), in the Ideal Cipher Model, $\mathcal{E}$ is sampled at random among all possible functions $\{0,1\}^\eta \to \{0,1\}^\eta$. Since y has never been queried to the block cipher, $x := \mathcal{E}(y)$ is indistinguishable from an independent random value, and so possess the same invariants as if $x \xleftarrow{\$} \mathcal{U}$ had been executed. Rules (B2) to (B5) simply preserve invariants that are unaffected by the computation of the block cipher on a value that has never been queried before.

Generic preservation rules.

The conditions for applying those rules, particularly $z \neq x, w, v$ were designed specifically so that the command would have no effect on the invariant. The invariant is therefore preserved.

As a result of all this, we have the following:

Proposition 1. *In the Ideal Cipher Model, the Hoare triples given in the previous rules are valid.*

As a result, our method can be used to prove the semantic security of an encryption mode by proving that, from the adversary's point of view, the ciphertexts are indistinguishable from random bits.

Proposition 2. *Let $\mathcal{E}_M(m_1|\dots|m_i, c_0|\dots|c_i)$: **var** $\boldsymbol{x_i}; \mathsf{c}_i$ be a generic encryption mode describe with our language, and let $IO = \{m_1, \dots, m_i, c_0, \dots, c_i\}$. If $\{\mathsf{true}\}\mathsf{c}_i \bigwedge_{k=0}^{i} \{\mathsf{Indis}(\nu c_k; IO)\}$ is valid for every i, then $\mathcal{E}_M$ is IND-CPA secure in the Ideal Cipher Model.*

We conclude with the following, which states that our method of proving security of encryption modes is sound in the standard model.

Proposition 3. *Let $\mathcal{E}_M$ be an encryption mode proven secure in the Ideal Cipher Model using the method of Proposition 2. If there exists a standard model algorithm A such that $\mathsf{Adv}_{A,M}^{ind-CPA}(\eta, X)$ is non-negligible, then there exists an algorithm B such that $\mathsf{Adv}_{B,\mathcal{E}}^{prf}$ is non-negligible.*

4 Examples

In this section we apply our method to the traditional encryption modes (CBC), (CFB), (OFB) and (CTR) in respectively Figure 3, 4, 5 and 6. For simplicity, we consider messages consisting of only 3 blocks. The reader can easily be convinced that the same invariant propagation holds for any finite number of blocks. In order to prove IND-CPA security of these encryption schemes we have to prove that $c_0 = IV, c_1, c_2, c_3$ are indistinguishable from random bitstrings when given $m_1, m_2, m_3, c_0, c_1, c_2$ and c_3. Of course our method fails in analyzing ECB encryption mode and the "counter" version of CBC, which are two insecure operation modes.

CBC & CFB : In Figure 3 and 4, we describe the application of our set of rules on CBC and CFB examples. The analysis of these two encryption modes are similar.

OFB : The order of the commands in our description of OFB may seem strange, but it is not without reason. The variable z_{i+1} must be computed before c_i because no rule can preserve the invariant $\mathsf{E}(\mathcal{E}, z_i)$ through the computation of c_i.

CTR : This scheme is the only one of the four encryption modes we have studied that uses the increment command. The analysis is presented in Figure 6. We can see how the *RCounter* invariant is used for proving the IND-CPA security of this mode.

5 Conclusion

We proposed an automatic method for proving the semantic security of symmetric encryption modes. We introduced a small programming language in order to describe these modes. We construct a Hoare logic to make assertions about variables and propagate the assertions with the execution of the commands in the language. If the program which represents an encryption mode satisfies some invariants at the end of our automatic analysis then we conclude that the encryption mode is IND-CPA secure.

Future work: An obvious extension to our work would be to add a loop construct to our grammar. This would remove the necessity of having a different program for each message length within a mode of operation. We are also considering an extension of our work to prove CCA security of encryption modes using approaches such as the one proposed in [10] or the method proposed in [8]. Another more complex and challenging direction is to propose an extended version of our Hoare Logic in order to be able to analyze "modern" encryption modes which use more complex mathematical operation or primitives, or to try to use our method to prove security properties of other block-cipher based construction, such as unforgeability for block-cipher based MACs, or collision-resistance for block-cipher based hash functions.

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge F(IV) \wedge \mathsf{E}(\mathcal{E}, IV)\}$	(R1)
$z_1 := IV \oplus m_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, z_1)\}$	(X2)(X4)
$c_1 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}) \wedge F(c_1)\}$	(B1)
$z_2 := c_1 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2) \wedge \mathsf{E}(\mathcal{E}, z_2)\}$	(X2)(X4)
$c_2 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}) \wedge F(c_2)\}$	(B1)
$z_3 := c_2 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_3) \wedge \mathsf{E}(\mathcal{E}, z_3)\}$	(X2)(X4)
$c_3 := \mathcal{E}(z_3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_3) \wedge Indis(\nu c_3; \mathsf{Var})\}$	(B1)

Fig. 3. Analysis of CBC encryption mode

$\mathcal{E}_{CFB}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV) \wedge F(IV) \wedge E(\mathcal{E}, IV)\}$	(R1)
$z_1 := \mathcal{E}(IV)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu z_1) \wedge F(z_1)\}$	(B1)(B2)
$c_1 := z_1 \oplus m_1$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, c_1)\}$	(G1)(X1)(X4)
$z_2 := \mathcal{E}(c_1)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge F(z_2)\}$	(B1)(B2)
$c_2 := z_2 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{E}(\mathcal{E}, c_2)\}$	(X1) (X4)
$z_3 := \mathcal{E}(c_2)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge F(z_3)\}$	(B1)
$c_3 := z_3 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2)$	(X1)
	$\wedge\ \mathsf{Indis}(\nu c_3; \mathsf{Var} - z_3)\}$	

Fig. 4. Analysis of CFB encryption mode

$\mathcal{E}_{OFB}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge F(IV) \wedge \mathsf{E}(\mathcal{E}, IV)\}$	(R1)
$z_1 := \mathcal{E}(IV)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \{F(z_1) \wedge \mathsf{E}(\mathcal{E}, z_1) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var})\}$	(B1)(B2)
$z_2 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var}) \wedge \mathsf{E}(\mathcal{E}, z_2)$	(B1)(B2)
	$\wedge\ F(z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var})\}$	
$c_1 := m_1 \oplus z_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, z_2)$	(G1)(G2)(X1)
	$\wedge\ F(z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var})\}\}$	(G4)
$z_3 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, z_3)$	(B1)(B2)
	$\wedge\ \mathsf{Indis}(\nu z_2; \mathsf{Var}) \wedge F(z_3) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})\}$	(B2)
$c_2 := m_2 \oplus z_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})$	(X1)
$c_3 := m_3 \oplus z_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu c_3; \mathsf{Var} - z_3)$	(X1)

Fig. 5. Analysis of OFB encryption mode

$\mathcal{E}_{CTR}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge F(IV) \wedge \mathsf{E}(\mathcal{E}, IV)\}$	(R1)
$ctr1 := IV + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1)$	(I3)
	$\wedge\ Rcounter(ctr1) \wedge \mathsf{E}(\mathcal{E}, ctr1)\}$	(I1)
$z_1 := \mathcal{E}(ctr1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1) \wedge Rcounter(ctr1)$	(B2)(B3)
	$\wedge\ F(z_1) \wedge \mathsf{E}(\mathcal{E}, z_1) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var})\}$	(B1)
$c_1 := m_1 \oplus z_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1) \wedge Rcounter(ctr1)$	(G1)(G3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(X1)
$ctr2 := ctr1 + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(I3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(G1)
	$\wedge\ Rcounter(ctr2) \wedge \mathsf{E}(\mathcal{E}, ctr2)\}$	(I2)
$z_2 := \mathcal{E}(ctr2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge Rcounter(ctr2)$	(B1)
	$\wedge\ F(z_2) \wedge \mathsf{E}(\mathcal{E}, z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var})\}$	(B3)
$c_2 := m_2 \oplus z_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge Rcounter(ctr2)$	(G3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2)\}$	(X1)
$ctr3 := ctr2 + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(I3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{E}(\mathcal{E}, ctr3)$	(I2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge Rcounter(ctr3)\}$	(G1)
$z_3 := \mathcal{E}(ctr3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(B1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge Rcounter(ctr3)$	(B3)
	$\wedge\ F(z_3) \wedge \mathsf{E}(\mathcal{E}, z_3) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})\}$	
$c_3 := m_3 \oplus z_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(G1)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(X1)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2)$	
	$\wedge\ \mathsf{Indis}(\nu c_3; \mathsf{Var} - z_3)\}$	

Fig. 6. Analysis of CTR encryption mode

References

1. Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:394, 1997.
2. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.
3. 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/`.
4. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In *FSE*, pages 389–407, 2004.
5. Debrup Chakraborty and Mridul Nandi. An improved security bound for HCTR. pages 289–302, 2008.
6. Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2006.
7. Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-counter-hash approach. *IEEE Transactions on Information Theory*, 54(4):1683–1699, 2008.
8. Judicael Courant, Marion Daubignard, Cristian Ene, Pascal Lafourcade, and Yassine Lahknech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, (CCS'08)*, Alexandria, USA, October 2008.
9. Judicaël Courant, Cristian Ene, and Yassine Lakhnech. Computationally sound typing for non-interference: The case of deterministic encryption. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 364–375. Springer, 2007.
10. Anand Desai. New paradigms for constructing symmetric encryption schemes secure against chosen-ciphertext attack. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 394–412, London, UK, 2000. Springer-Verlag.
11. Martin Gagné, Pascal Lafourcade, Yassine Lakhnech, and Reihaneh Safavi-Naini. Automated security proof for symmetric encryption modes. Manuscript,2009. Available at `http://pages.cpsc.ucalgary.ca/~mgagne/TR_Asian.pdf`,.
12. Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2004.
13. Shai Halevi. Invertible universal hashing and the tet encryption mode. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007.
14. Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2003.
15. Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.

16. Tetsu Iwata and Kaoru Kurosawa. OMAC: One-key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.
17. Tetsu Iwata and Kaoru Kurosawa. On the security of a new variant of OMAC. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2003.
18. Tetsu Iwata and Kaoru Kurosawa. Stronger security bounds for OMAC, TMAC, and XCBC. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT*, volume 2904 of *Lecture Notes in Computer Science*, pages 402–415. Springer, 2003.
19. Charanjit S. Jutla. Encryption modes with almost free message integrity. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 529–544, London, UK, 2001. Springer-Verlag.
20. Kaoru Kurosawa and Tetsu Iwata. TMAC: Two-key CBC MAC. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2003.
21. Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit - a new construction. In *Fast Software Encryption Ç02, Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 2001.
22. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 31–46, London, UK, 2002. Springer-Verlag.
23. David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (XCB) mode of operation, 2007.
24. David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In *INDOCRYPT*, pages 343–355, 2004.
25. Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.
26. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 196–205, New York, NY, USA, 2001. ACM.
27. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. http://eprint.iacr.org/2004/332.
28. Peng Wang, Dengguo Feng, and Wenling Wu. On the security of tweakable modes of operation: TBC and TAE. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 274–287. Springer, 2005.

Formal Indistinguishability extended to the Random Oracle Model

Cristian Ene, Yassine Lakhnech and Van Chan Ngo *

Université Grenoble 1, CNRS,Verimag

Abstract. Several generic constructions for transforming one-way functions to asymmetric encryption schemes have been proposed. One-way functions only guarantee the weak secrecy of their arguments. That is, given the image by a one-way function of a random value, an adversary has only negligible probability to compute this random value. Encryption schemes must guarantee a stronger secrecy notion. They must be at least resistant against indistinguishability-attacks under chosen plaintext text (IND-CPA). Most practical constructions have been proved in the random oracle model. Such computational proofs turn out to be complex and error prone. Bana et al. have introduced *Formal Indistinguishability Relations, (FIR for short)* as an appropriate abstraction of computational indistinguishability. In this paper, we revisit their work and extend the notion of FIR to cope with the random oracle model on one hand and adaptive adversaries on the other hand. Indeed, when dealing with hash functions in the random oracle model and one-way functions, it is important to correctly abstract the notion of weak secrecy. Moreover, one needs to extend frames to include adversaries in order to capture security notions as IND-CPA. To fix these problems, we consider pairs of formal indistinguishability relations and *formal non-derivability relations*. We provide a general framework along with general theorems, that ensure soundness of our approach and then we use our new framework to verify several examples of encryption schemes among which the construction of Bellare Rogaway and Hashed ElGamal.

1 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 *provably correct* cryptographic systems (e.g., primitives and protocols). There are two main frameworks for analyzing cryptographic systems; the *symbolic framework*, originating from the work of Dolev and Yao [15], and the *computational approach*, growing out of the work of [17]. A significant amount of effort has been made in order to link both approaches and profit from the advantages of each of them. Indeed, while the symbolic approach is more amenable to automated proof methods, the computation approach can be more realistic.

In their seminal paper [1] Abadi and Rogaway investigate the link between the symbolic model on one hand and the computational model on the other hand. More precisely, they introduce an equivalence relation on terms and prove that equivalent terms correspond to indistinguishable distributions ensembles, when interpreted in the computational model. The work of Abadi and Rogaway has been extended to active adversaries and various cryptographic primitives in e.g. [20, 19, 14, 18]. An other line of work, also considering active adversaries is followed by Backes, Pfitzmann and Waidner using *reactive simulatability* [6, 5] and Canetti [12, 13] using *universal composability*.

* Grenoble, email:name@imag.fr This work has been partially supported by the ANR projects SCALP, AVOTE and SFINCS

Related works A recently emerging branch of relating symbolic and computational models for passive adversaries is based on *static equivalence* from π-calculus [3], induced by an *equational theory*. Equational theories provide a framework to specify algebraic properties of the underlying signature, and hence, symbolic computations in a similar way as for abstract data types. That is, for a fixed equational theory, a term describes a computation in the symbolic model. Thus, an adversary can distinguish two terms, if he is able to come up with two computations that yield the same result when applied to one term but different results when applied to the other term. Such a pair of terms is called a *test*. This idea can be extended to *frames*, which roughly speaking are tuples of terms. Thus, a *static equivalence* relation is fully determined by the underlying equational theory, as two frames are *statically equivalent*, if there is no test that separates them. In [9] Baudet, Cortier and Kremer study soundness and faithfulness of static equivalence for general equational theories and use their framework to prove soundness of exclusive or as well as certain symmetric encryptions. Abadi et al. [2] use static equivalence to analyze of guessing attacks.

Bana, Mohassel and Stegers [8] argue that even though static equivalence works well to obtain soundness results for the equational theories mentioned above, it does not work well in other important cases. Consider for instance the Decisional Diffie Hellman assumption (DDH for short) that states that the tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), where a, b, c are randomly sampled, are indistinguishable. It does not seem to be obvious to come up with an equational theory for group exponentiation such that the induced static equivalence includes this pair of tuples without including others whose computational indistinguishability is not proved to be a consequence of the DDH assumption. The static equivalence induced by the equational theory for group exponentiation proposed in [9] includes the pair (g, g^a, g^b, g^{a^2b}) and (g, g^a, g^b, g^c). It is unknown whether the computational indistinguishability of these two distributions can be proved under the DDH assumption. Therefore, Bana et al. propose an alternative approach to build symbolic indistinguishability relations and introduce *formal indistinguishability relations (FIR)*. A FIR is defined as a closure of an initial set of equivalent frames with respect to simple operations which correspond to steps in proofs by reduction. This leads to a flexible symbolic equivalence relation. FIR has nice properties. In order to prove soundness of a FIR it is enough to prove soundness of the initial set of equivalences. Moreover, static equivalence is one instance of a FIR. Bana et al. show that it is possible to come up with a FIR whose soundness is equivalent to the DDH assumption.

Contributions. In this paper, we extend Bana et al.'s approach by introducing a notion of symbolic equivalence that allows us to prove security of encryption schemes symbolically. More specifically, we would like to be able to treat generic encryption schemes that transform one-way functions to IND-CPA secure encryption schemes. Therefore, three problems need to be solved. First, we need to cope with one-way functions. This is another example where static equivalence does not seem to be appropriate. Indeed, let f be a one-way function, that is, a function that is easy to compute but difficult to invert. It does not seem easy to come with a set of equations that capture the one-wayness of such a function. Consider the term $f(a|b)$, where $|$ is bit-string concatenation. If f is a one-way function then we know that we cannot easily compute $a|b$ given $f(a|b)$ for uniformly sampled a and b. However, nothing prevents us from being able to compute a for instance. Introducing equations that allow us to compute a from $f(a|b)$, e.g., $g(f(a|b)) = a$, may exclude some one-way functions and does not solve the problem. For instance, nothing prevents us from computing a prefix of b (its first half for instance), a

prefix of the prefix, etc..... . The second problem that needs to be solved is related to the fact that almost all practical provably secure encryption schemes are analyzed in the random oracle model. The random oracle model is an idealized model in which hash functions are randomly sampled functions. In this model, adversaries have oracle access to these functions. An important property is that if an adversary is unable to compute the value of an expression a and if $H(a)$ has not been leaked then $H(a)$ looks like a uniformly sampled value. Thus, we need to be able to symbolically prove that a value of a given expression a cannot be computed by any adversary. This is sometimes called *weak secrecy* in contrast to indistinguishability based secrecy. To cope with this problem, our notion of symbolic indistinguishability comes along with a *non-derivability* symbolic relation. Thus in our approach, we start from an initial pair of a non-derivability relation and a frame equivalence relation. Then, we provide rules that define a closure of this pair of relations in the spirit of Bana et al.'s work. Also in our case, soundness of the obtained relations can be checked by checking soundness of the initial relations. The third problem is related to the fact that security notions for encryption schemes such IND-CPA and real-or-random indistinguishability of cipher-text under chosen plaintext involve a generated from of *active* adversaries. Indeed, these security definitions correspond to two-phase games, where the adversary first computes a value, then a challenge is produced, the the adversary tries to solve the challenge. Static equivalence and FIR (as defined in [8]) consider only passive adversaries. To solve this problem we consider frames that include variables that correspond to adversaries. As frames are finite terms, we only have finitely many such variables. This is the reason why we only have a degenerate form of active adversaries which is enough to treat security of encryption schemes and digital signature, for instance.

The closure rules we propose in our framework are designed with the objective of minimizing the initial relations which depend on the underlying cryptographic primitives and assumptions.

We illustrate the framework by considering security proofs of the construction of Bellare and Rogaway [11] and Hash El Gamal [7].

Outline of the paper. In Section 2, we introduce the symbolic model used for describing generic asymmetric encryption schemes. In Section 3, we describe the computational framework and give definitions that relate the two models. In Section 4, we introduce our definition of formal indistinguishability relation and formal non-derivability relation. We also present our method for proving IND-CPA security. In Section 5, we illustrate our framework: we prove the construction of Bellare and Rogaway [11] and Hash El Gamal [7], and we give a sketch of the proof of encryption scheme proposed by Pointcheval in [23]. Finally, in Section 7 we conclude.

2 Symbolic semantics

2.1 Terms and subsitutions

A *signature* $\Sigma = (\mathcal{S}, \mathcal{F}, \mathcal{H})$ consists of a countably infinite set of *sorts* $\mathcal{S} = \{s, s_1, ...\}$, a finite set of *function symbols*, $\mathcal{F} = \{f, f_1, ...\}$, and a finite set of *oracle symbols*, $\mathcal{H} = \{g, h, h_1, ...\}$ together with arities of the form $ar(f)$ or $ar(h) = s_1 \times ... \times s_k \to s, k \geq 0$. Symbols in $\mathcal{F}$ that take $k = 0$ as arguments are called *constants*. We suppose that there are three pairwise disjoint sets $\mathcal{N}$, $\mathcal{X}$ and $\mathcal{P}$. $\mathcal{N}$ is the set of names, $\mathcal{X}$ is the set of first-order variables, and $\mathcal{P}$ is the set of second order variables. We assume that both names and variables are sorted, that is, to each name or variable u, a sort $\mathbf{s}$ is assigned; we use $\mathbf{s}(s)$ for the sot of u. Variables $p \in \mathcal{P}$

have arities $ar(p) = \mathbf{s}_1 \times ... \times \mathbf{s}_k \rightarrow \mathbf{s}$. We suppose that there are a countable number of names, variables and p-variables for each sort or arity.

A renaming is a bijection $\tau : \mathcal{N} \rightarrow \mathcal{N}$ such that $\mathbf{s}(a) = \mathbf{s}(\tau(a))$. As usual, we extend the notation $\mathbf{s}(T)$ to denote the sort of a term T. Terms of sort $\mathbf{s}$ are defined by the grammar:

$T ::=$		*term of sort s*
	$\|x$	*variable x of sort* $\mathbf{s}$
	$\|p(T_1, \ldots, T_k)$	*variable p of arity* $\mathbf{s}(T_1) \times ... \times \mathbf{s}(T_k) \rightarrow \mathbf{s}$
	$\|n$	*name n of sort* $\mathbf{s}$
	$\|f(T_1, \ldots, T_k)$	*application of symbol* $f \in \mathcal{F}$ *with arity* $\mathbf{s}(T_1) \times ... \times \mathbf{s}(T_k) \rightarrow \mathbf{s}$
	$\|h(T_1, \ldots, T_k)$	*call of hash-function* $h \in \mathcal{H}$ *with arity* $\mathbf{s}(T_1) \times ... \times \mathbf{s}(T_k) \rightarrow \mathbf{s}$

We use $fn(T)$, $pvar(T)$ and $var(T)$ for the set of free names, the set of p-variables and the set of variables that occur in the term T, respectively. We use meta-variables u, v, w to range over names and variables. We use $st(T)$ for the set of sub-terms of T, defined in the usual way: $st(u) \stackrel{def}{=} \{u\}$ if u is a name or a variable, and $st(l(T_1, \ldots, T_k)) \stackrel{def}{=} \{l(T_1, \ldots, T_k)\} \bigcup_{i \in \{1,...k\}} st(T_i)$, if $l \in \mathcal{F} \cup \mathcal{H} \cup \mathcal{P}$. A term T is closed if and only if it does not have any free variables (but it may contain p-variables, names and constant symbols), that means $var(T) = \emptyset$. The set of terms is denoted by $\mathbf{T}$.

Symbols in $\mathcal{F}$ are intended to model cryptographic primitives, symbols in $\mathcal{H}$ are intended to model cryptographic oracles (in particular, hash functions in the ROM model), whereas names in $\mathcal{N}$ are used to model secrets, that is, concretely random numbers. Variables $p \in \mathcal{P}$ are intended to model queries and challenges made by adversaries (they can depend on previous queries).

Definition 1 (Substitution). *A substitution σ is a mapping from variables to terms whose domain is finite and such that $\sigma(x) \neq x$, for each x in the domain. A substitution σ is written $\sigma = \{x_1 = T_1, ..., x_n = T_n\}$, where $dom(\sigma) = \{x_1, ..., x_n\}$ is its domain.*

We only consider *well-sorted* substitutions for which x_i and T_i have the same sort, $var(T_i) \subseteq \{x_1, \ldots, x_n\}$ and there is no circular dependence $x_{i_1} = T_{i_1}(\ldots x_{i_2} \ldots), x_{i_2} = T_{i_2}(\ldots x_{i_3} \ldots), \ldots x_{i_k} = T_{i_k}(\ldots x_{i_1} \ldots)$. A substitution is called *closed* if all terms T_i are closed. We let $var(\sigma) = \cup_i var(T_i)$, $pvar(\sigma) = \cup_i pvar(T_i)$, $n(\sigma) = \cup_i fn(T_i)$, and extend the notations $pvar(.)$, $var(.)$, $n(.)$ and $st(.)$ to tuples and set of terms and substitutions in the obvious way. The application of a substitution σ to a term T is written as $\sigma(T) = T\sigma$. Let $\sigma = \{x_1 = T_1, ..., x_n = T_n\}$ and $\sigma' = \{x'_1 = T'_1, ..., x'_m = T'_m\}$ be substitutions such that $dom(\sigma) \cap dom(\sigma') = \emptyset$. Then, $\sigma|\sigma'$ denotes the sunbstitution $\{x_1 = T_1, ..., x_n = T_n, x'_1 = T'_1, ..., x'_m = T'_m\}$.

The abstract semantics of symbols is described by an equational theory E, that is an equivalence (denoted as $=_E$) which is stable with respect to application of contexts and well-sorted substitutions of variables. We further require that E is stable under renamings.

Definition 2 (Equational Theory.). *An equational theory for a given signature is an equivalence relation $E \subseteq \mathcal{T} \times \mathcal{T}$ (written as $=_E$ in infix notation) on the set of terms such that*

1. *$T_1 =_E T_2$ implies $T_1\sigma =_E T_2\sigma$ for every substitution σ;*
2. *$T_1 =_E T_2$ implies $T\{x = T_1\} =_E T\{x = T_2\}$ for every term T and every variable x;*
3. *$T_1 =_E T_2$ implies $\tau(T_1) =_E \tau(T_2)$ for every renaming τ.*

All definitions from now on are given in the context of an (implicit) equational theory E.

Exemples. For instance, symmetric and deterministic encryption can be modeled by the theory E_{enc} generated by the classical equation $E_{enc} = \{dec(enc(x,y),y) =_{E_{enc}} x\}$. A trapdoor one-way function can be modeled by the theory E_{ow} generated by the equation $E_{ow} = \{f^{-1}(f(x,pub(sk)),sk) =_E x,\}$, where sk is the secret key (the trapdoor), f^{-1} is the inverse function of the trapdoor one-way function f, and $pub(sk)$ is the public information, respectively.

2.2 Frames

Frames ([4]) represent sequences of messages (or pieces of information) observed by an adversary. Formally:

Definition 3 (Frame). *A frame is an expression of the form $\phi = \nu\widetilde{n}.\sigma$ where σ is a well-sorted substitution, and $\widetilde{n}$ is $n(\sigma)$, the set of all names occuring in σ. By abus of notation we also use $n(\phi)$ for $\widetilde{n}$, the set of names bounded in the frame ϕ.*

The novelty of our definition of frames consists in permitting adversaries to interact with frames using p-variables. This is necessary to be able to cope with adaptive adversaries. We note the set of frames by **F**.

Next, we define composition and parallel composition of frames. Let $\phi = \nu\widetilde{n}.\{x_1 = T_1, ..., x_n = T_n\}$ and $\phi' = \nu\widetilde{n'}.\sigma$ be frames with $\widetilde{n} \cap \widetilde{n'} = \emptyset$. Then, $\phi\phi'$ denotes the frame $\nu(\widetilde{n} \cup \widetilde{n'}).\{x_1 = T_1\sigma, ..., x_n = T_n\sigma\}$. Let now $\phi_1 = \nu\widetilde{n_1}.\sigma_1, ..., \phi_k = \nu\widetilde{n_k}.\sigma_k$ be frames with pairwisely disjoint domains and pairwisely disjoint bounded names $\widetilde{n_i}$. Their *parallel composition*, $\{\phi_1|\phi_2|...|\phi_n\}$ is the frame $\nu(\bigcup_{i=1}^{k}\widetilde{n_i}).\sigma_1|\cdots|\sigma_k$. The *iteration* of a frame ϕ is the iterative composition of ϕ with itself until it remains unchanged : $\phi^* = (\ldots((\phi)\phi)\ldots)\phi$.

Definition 4 (Static equivalence). *Let ϕ and ϕ' be two frames such that $\phi^* = \nu\widetilde{n}.\sigma$ and $\phi'^* = \nu\widetilde{n}.\sigma'$ with $\sigma = \{x_1 = T_1, ..., x_n = T_n\}$ and $\sigma' = \{x_1 = T'_1, ..., x_n = T'_n\}$. Given the equational theory E, we say that ϕ and ϕ' are statically equivalent written $\phi =_E \phi'$, if and only if $T_i\sigma =_E T'_i\sigma'$ for all i.*

Some obvious properties: $\phi =_E \phi'$ implies $\psi\phi =_E \psi\phi'$ and $\tau(\phi) =_E \tau(\phi')$ for any frames ϕ, ϕ' and ψ and any renaming τ.

3 Computational Semantics

3.1 Distributions and indistinguishability

Let us note $\eta \in \mathbb{N}$ the security parameter. We are interested in analyzing generic schemes for asymmetric encryption assuming ideal hash functions. That is, we are working in the *random oracle model* [16, 11]. Using standard notations, we write $h \xleftarrow{r} \Omega$ to denote that h is randomly chosen from the set of functions with appropriate domain (deppendong on η). By abuse of notation, for a list $\boldsymbol{H} = h_1, \cdots, h_m$ of hash functions, we write $\boldsymbol{H} \xleftarrow{r} \Omega$ instead of the sequence $h_1 \xleftarrow{r} \Omega, \ldots, h_m \xleftarrow{r} \Omega$. We fix a finite set $\mathcal{H} = \{h_1, \ldots, h_n\}$ of hash functions. We assume an arbitrary but fixed ordering on $\mathcal{H}$; just to be able to switch between set-based and vector-based notation. A *distribution ensemble* is a countable sequence of distributions $\{X_\eta\}_{\eta\in\mathbb{N}}$. We only consider distribution ensembles that can be constructed in polynomial time by probabilistic algorithms that have oracle access to $\mathcal{O} = \mathcal{H}$. Given two distribution ensembles $X = \{X_\eta\}_{\eta\in\mathbb{N}}$

and $X' = \{X'_\eta\}_{\eta\in\mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A}, \eta, X, X') = \mathsf{Pr}[x \xleftarrow{r} X_\eta : \mathcal{A}^{\mathcal{O}}(\eta, x) = 1] - \mathsf{Pr}[x \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathcal{O}}(\eta, x) = 1].$$

Then, two distribution ensembles X and X' are called *indistinguishable* (denoted by $X \sim X'$) if for any probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}(\mathcal{A}, \eta, X, X')$ is negligible as a function of η, that is, for any $n > 0$, it become eventually smaller than η^{-n} as η tends to infinity. We insist that all security notions we are going to use are in the ROM, where all algorithms, including adversaries, are equipped with oracle access to the hash functions.

3.2 Frames as distributions

We now give terms and frames a computational semantics parameterized by a computable implementation of the primitives in the random oracle model. Provided a set of sorts $\mathcal{S}$ and a set of symbols $\mathcal{F}$, a *computational algebra* $A = (\mathcal{S}, \mathcal{F})$ consists of

- a sequence of non-empty finite set of bit strings $[\![s]\!]_A = \{[\![s]\!]_{A,\eta}\}_{\eta\in\mathbb{N}}$ with $[\![s]\!]_{A,\eta} \subseteq \{0,1\}^*$ for each sort $s \in S$. For simplicity of the presentation, we assume that all sorts are large domains, whose cardinalities are exponential in the security parameter η;
- a sequence of polynomial time computable functions $[\![f]\!]_A = \{[\![f]\!]_{A,\eta}\}_{\eta\in\mathbb{N}}$ with $[\![f]\!]_{A,\eta} : [\![s_1]\!]_{A,\eta} \times ... \times [\![s_k]\!]_{A,\eta} \to [\![s]\!]_{A,\eta}$ for each $f \in \mathcal{F}$ with $ar(f) = s_1 \times ... \times s_k \to s$;
- a polynomial time computable congruence $=_{A,\eta,s}$ for each sort s, in order to check the equality of elements in $[\![s]\!]_{A,\eta}$ (the same element may be represented by different bit strings). By congruence, we mean a reflexive, symmetric, and transitive relation such that $e_1 =_{A,s_1,\eta} e'_1, ..., e_k =_{A,s_k,\eta} e'_k \Rightarrow [\![f]\!]_{A,\eta}(e_1, ..., e_k) =_{A,s,\eta} [\![f]\!]_{A,\eta}(e'_1, ..., e'_k)$ (we usually omit s,η and A and write $=$ for $=_{A,s,\eta}$);
- a polynomial time procedure to draw random elements from $[\![s]\!]_{A,\eta}$; we denote such a drawing by $x \leftarrow^R [\![s]\!]_{A,\eta}$; for simplicity, in this paper we suppose that all these drawing follow a uniform distribution.

From now on we assume a fixed computational algebra $(\mathcal{S}, \mathcal{F})$, and a fixed η, and for simplicity we omit the indices A,s and η.

Given $\mathcal{H}$ a fixed set of hash functions, and $(\mathcal{A}_i)_{i\in I}$ a fixed set of polynomial-probabilistc functions (can be seen as a polynomial-probabilistc adversary $\mathcal{A}^{\mathcal{O}}$ that takes an additional input i), we associate to each frame $\phi = \nu\widetilde{n}.\{x_1 = T_1, \ldots, x_k = T_k\}$ a sequence of distributions $[\![\phi]\!]_{\mathcal{H},\mathcal{A}}$ computed as follows:

- for each name n of sort s appearing in $\widetilde{n}$, draw a value $\hat{n} \xleftarrow{r} [\![s]\!]$;
- for each variable $x_i (1 \le i \le k)$ of sort s_i, compute $\hat{T}_i \in [\![s_i]\!]$ recursively on the structure of terms: $\hat{x_i} = \hat{T}_i$;
- for each call $h_i(T'_1, \ldots, T'_m)$ compute recursively on the structure of terms: $\widehat{h_i(T'_1, \ldots, T'_m)} = h_i(\hat{T}'_1, \ldots, \hat{T}'_m)$;
- for each call $f(T'_1, \ldots, T'_m)$ compute recursively on the structure of terms: $\widehat{f(T'_1, \ldots, T'_m)} = [\![f]\!](\hat{T}'_1, \ldots, \hat{T}'_m)$;
- for each call $p_i(T'_1, \ldots, T'_m)$ compute recursively on the structure of terms and draw a value $\widehat{p_i(T'_1, \ldots, T'_m)} \xleftarrow{r} \mathcal{A}^{\mathcal{O}}(i, \hat{T}'_1, \ldots, \hat{T}'_m)$;

– return the value $\hat{\phi} = \{x_1 = \hat{T_1}, \ldots, x_k = \hat{T_k}\}$.

Such values $\phi = \{x_1 = bse_1, \ldots, x_n = bse_n\}$ with $bse_i \in [\![s_i]\!]$ are called *concrete frames*. We extend the notation $[\![.]\!]$ to (sets of) closed terms in the obvious way. We also generalize the notation to terms or frames with free variables and free names, by specifying the concrete values for all of them: $[\![.]\!]_{\{n_1=bsn_1,\ldots n_k=bsn_k,x_1=bse_1,\ldots,x_l=bse_l\}}$.

Now the concrete semantics of a frame ϕ with respect to an adversary $\mathcal{A}$, is given by the following sequence of distributions (one for each implicit η):

$$[\![\phi]\!]_{\mathcal{A}} = \left[\mathcal{H} \stackrel{r}{\leftarrow} \Omega; \mathcal{O} = \mathcal{H}; \hat{\phi} \stackrel{r}{\leftarrow} [\![\phi]\!]_{\mathcal{H},\mathcal{A}} : \hat{\phi}\right]$$

When $pvar(\phi) = \emptyset$, the concrete semantics of ϕ does not depend on the adversary $\mathcal{A}$ and we will use the notation $[\![\phi]\!]$ (or $[\![\phi]\!]_{\mathcal{H}}$) instead of $[\![\phi]\!]_{\mathcal{A}}$ (respectively $[\![\phi]\!]_{\mathcal{H},\mathcal{A}}$).

3.3 Soundness and Completeness

The computational model of a cryptographic scheme is closer to reality than its formal representation by being a more detailed description. Therefore, the accuracy of a formal model can be characterized based on how close it is to the computational model. For this reason, we introduce the notions of soundness and completeness that relate relations in the symbolic model with respect to similar relations in the computational model. Let E be an equivalence theory and let $R_1 \subseteq \mathbf{T} \times \mathbf{T}$, $R_2 \subseteq \mathbf{F} \times \mathbf{T}$, and $R_3 \subseteq \mathbf{F} \times \mathbf{F}$ be relations on closed frames, on closed terms, and relations on closed frames and terms, respectively.

- Then R_1 is $=$-sound iff for every closed terms T_1, T_2 of the same sort, $(T_1, T_2) \in R_1$ implies that $\mathsf{Pr}[\hat{e_1}, \hat{e_2} \stackrel{r}{\leftarrow} [\![T_1, T_2]\!]_{\mathcal{A}} : \hat{e}_1 \neq \hat{e}_2))]$ is negligible for any polynomial time adversary $\mathcal{A}$.
- Then R_1 is $=$-complete iff for every closed terms T_1, T_2 of the same sort, $(T_1, T_2) \notin R_1$ implies that $\mathsf{Pr}[\hat{e_1}, \hat{e_2} \stackrel{r}{\leftarrow} [\![T_1, T_2]\!]_{\mathcal{A}} : \hat{e}_1 \neq \hat{e}_2))]$ is non-negligible for some polynomial time adversary $\mathcal{A}$.
- Then R_2 is $\not\vdash$-sound iff for every closed frame ϕ and term T, $(\phi, T) \in R_2$ implies that $\mathsf{Pr}[\hat{\phi}, \hat{e} \stackrel{r}{\leftarrow} [\![\phi, T]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}]$ is negligible for any probabilistic polynomial-time adversary $\mathcal{A}$.
- Then R_2 is $\not\vdash$-complete iff for every closed frame ϕ and term T, $(\phi, T) \notin R_2$ implies that $\mathsf{Pr}[\hat{\phi}, \hat{e} \stackrel{r}{\leftarrow} [\![\phi, T]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}]$ is non-negligible for some polynomial-time adversary $\mathcal{A}$.
- Then R_3 is $\approx_E$-sound iff for every frames ϕ_1, ϕ_2 with the same domain, $(\phi_1, \phi_2) \in R_3$ implies that $([\![\phi_1]\!]_{\mathcal{A}}) \sim ([\![\phi_2]\!]_{\mathcal{A}})$ for any probabilistic polynomial-time adversary $\mathcal{A}$.
- Then R_3 is $\approx_E$-complete iff for every frames ϕ_1, ϕ_2 with the same domain, $(\phi_1, \phi_2) \notin R_3$ implies that $([\![\phi_1]\!]_{\mathcal{A}}) \not\sim ([\![\phi_2]\!]_{\mathcal{A}})$ for some probabilistic polynomial-time adversary $\mathcal{A}$.

4 Formal relations

One challenge of the paper is to propose appropriate symbolic relations that correctly abstract computational properties as indistinguishability of two distributions or weak secrecy of some random value (that is, the adversary has only negligible probability to compute it). In this section we provide two symbolic relations (called formal indistinguishability relation and formal non-derivability relation) that are sound abstractions for the two above computational properties.

First we define well-formed relations and we recall a simplified definition of a formal indistinguishability relation as proposed in [8].

Definition 5 (Well-formed relations). *A relation $S_d \subseteq \boldsymbol{F} \times \boldsymbol{T}$ is called* **well-formed** *if $fn(M) \subseteq n(\phi)$ for any $(\phi, M) \in S_d$, and a relation $S_i \subseteq \boldsymbol{F} \times \boldsymbol{F}$ is* **well-formed** *if $dom(\phi_1) = dom(\phi_2)$ for any $(\phi_1, \phi_2) \in S_i$.*

Definition 6. *[FIR [8]] A well-formed equivalence relation $\cong \subseteq \boldsymbol{F} \times \boldsymbol{F}$ is called a* **formal indistinguishability relation (FIR for short)** *with respect to the equational theory $=_E$, if $\cong$ is closed with respect to the following closure rules:*
(GE1) If $\phi_1 \cong \phi_2$ then $\phi\phi_1 \cong \phi\phi_2$, for any frame ϕ such that $var(\phi) \subseteq dom(\phi_i)$ and $n(\phi) \cap n(\phi_i) = \emptyset$.
(GE2) $\phi \cong \phi'$ for any frame ϕ' such that $\phi' =_E \phi$.
(GE3) $\tau(\phi) \cong \phi$ for any renaming τ.

This definition is a good starting point to capture indistinguishability in the following sense: if we have a correct implementation of the abstract algebra (i.e. $=_E$ is $=$-sound) and we were provided with some initial relation S (reflecting some computational assumption) which is $\approx$-sound , then the closure of S using the above rules produces a larger relation which still remains $\approx$-sound. But in order to use this definition for real cryptographic constructions , we need to enrich it in several aspects. First, most of constructions which are proposed in the literature, ([10], [27], [21], [23], [25], [11]) use bijective functions (XOR-function or trapdoor permutation) as basic bricks. To deal with these constructions, we add the following closure rule:
(GE4) If M, N are terms such that $N[M/z] =_E y$, $M[N/y] =_E z$ and $var(M) = \{y\}$ and $var(N) = \{z\}$, then for any substitution σ such that $r \notin (fn(\sigma) \cup fn(M) \cup fn(N))$ and $x \notin dom(\sigma)$ it holds $\nu\widetilde{n}.r.\{\sigma, x = M[r/y]\} \cong \nu\widetilde{n}.r.\{\sigma, x = r\}$.

Second, cryptographic constructions use often hash functions. In ideal models, hash functions are primitives that if applied to a weakly secret argument, produce a completely random value (modeled by random functions [11] or by pseudo-random permutations [22]). And they are quite frequent primitives in cryptography that only ensure weak secrecy. For instance one-way functions only guarantee that an adversary that possesses the image by a one-way function of a random value, has only a negligible probability to compute this value. The computational Diffie-Hellman (CDH) assumption states that if given the tuple g, g^a, g^b for some randomly-chosen generator g and some random values a, b, it is computationally intractable to compute the value g^{a*b} (equivalently g^{a*b} is a weakly secret value). This motivates us to introduce the **formal non-derivability relation** as an abstraction of weak secrecy. Let us explain the basic closure rules of this relation. Since we assume that all sorts will be implemented by large finite sets of bit strings, it is clearly that
(GD1) $\nu r.\emptyset \not\vdash r$.

Next rule captures the fact that renaming does not change the concrete semantics of terms or frames.
(GD2) If $\phi \not\vdash M$ then $\tau(\phi) \not\vdash \tau(M)$ for any renaming τ.

If the equational theory is preserved in the computational world, then equivalent terms or frames are indistinguishable.
(GD3) If $\phi \not\vdash M$ then $\phi \not\vdash N$ for any term $N =_E M$.
(GD4) If $\phi \not\vdash M$ then $\phi' \not\vdash M$ for any frame $\phi' =_E \phi$.

If some bit string (concrete implementation of some symbolic term M) is weakly secret, then all polynomially computation (abstracted by the symbolic frame ϕ') does not change this.

(GD5) If $\phi \not\models M$ then $\phi'\phi \not\models M$ for any frame ϕ' such that $var(\phi') \subseteq dom(\phi)$ and $n(\phi') \cap bn(\phi) = \emptyset$.

Next rule establishes a relationship between indistiguishability and secrecy: if two distributions are indistinguishable, then they leak exactly the same information.
(GD6) If T, U are terms such that $U[T/y] =_E z$ and $z \in var(T) \setminus var(U)$ and $(fn(T) \cup fn(U)) \cap \widetilde{n} = \emptyset$, then for all substitutions σ_1, σ_2 such that $x \notin dom(\sigma_i)$ and $\nu\widetilde{n}.\{\sigma_1, x = T[M/z]\} \cong \nu\widetilde{n}.\{\sigma_2, x = T[N/z]\}$ and $\nu\widetilde{n}.\sigma_1 \not\models M$ then $\nu\widetilde{n}.\sigma_2 \not\models N$.

And now the rule that captures the power of hash functions in the Random Oracle Model: the image by a random function of a weakly secret value is a completely random value.
(HE1) If $\nu\widetilde{n}.r.\sigma[r/h(T)] \not\models T$ and $r \notin n(\sigma)$, then $\nu\widetilde{n}.\sigma \cong \nu\widetilde{n}.r.\sigma[r/h(T)]$.

The following definition formalizes the tight connection between FIR and FNDR.

Definition 7 (FNDR and FIR). *A pair of well formed relations* $(\not\models, \cong)$ *is a pair of* **(formal non-derivability relation, formal indistinguishability relation)** *with respect to the equational theory* $=_E$*, if* $(\not\models, \cong)$ *is closed with respect to the rules (GD1), ..., (GD6),(GE1),...,(GE4), (HE1) and* $\cong$ *is an equivalence.*

The following theorem shows that if a pair of FIR and FNDR relations was generated by the initial sets $S_d \subseteq \mathbf{F} \times \mathbf{T}$ and $S_i \subseteq \mathbf{F} \times \mathbf{F}$, then it is sufficient to check only soundness of elements in S_d and S_i to ensure that the closures $\langle S_d \rangle_{\not\models}$ and $\langle S_i \rangle_{\cong}$ are sound. We define $(D_1, I_1) \sqsubset (D_2, I_2)$ if and only if $D_1 \subseteq D_2$ and $I_1 \subseteq I_2$. It is easy to see that $\sqsubset$ is an order.

Theorem 1. *Let* (S_d, S_i) *be a well-formed pair of relations. Then, it exists a unique smallest (with respect to* $\sqsubset$*) pair denoted* $(\langle S_d \rangle_{\not\models}, \langle S_i \rangle_{\cong})$ *of (FNDR, FIR) such that* $\langle S_d \rangle_{\not\models} \supseteq S_d$ *and* $\langle S_i \rangle_{\cong} \supseteq S_i$*. In addition, if* $=_E$ *is* $=$*-sound,* S_d *is* $\not\vdash$*-sound and* S_i *is* $\approx$*-sound, then also* $\langle S_d \rangle_{\not\models}$ *is* $\not\vdash$*-sound and* $\langle S_i \rangle_{\cong}$ *is* $\approx$*-sound.*

5 Applications

We apply the framework of Section 4 in order to prove IND-CPA security of several generic constructions for asymmetric encryptions. So we will consider pairs of relations $(\not\models, \cong) = (\langle S_d \rangle_{\not\models}, \langle S_i \rangle_{\cong})$ generated by some initial sets (S_d, S_i), in different equational theories. We assume that all $=_E$, S_d, S_i that are considerd in this section satisfy the conditions of Theorem 1. We emphasize the following fact: adding other equations than those considered does not break the computational soundness of results proved in this section, as long as the computational hypothesis encoded by S_d and S_i still hold.
First we introduce a general abstract algebra, and then we will extend it to cover different constructions. We consider three sorts $Data$, $Data^1$, $Data^2$, and the symbols $|| : Data^1 \times Data^2 \rightarrow Data$, $\oplus_S : S \times S \rightarrow S$, $0_S : S$, with $S \in \{Data, Data^1, Data^2\}$ and $\pi_j : Data \rightarrow Data^j$, with $j \in \{1, 2\}$. For simplicity, we omit the indice S when using $\oplus_S$ or 0_S . The equational theory E_g is generated by:
(XEq1) $x \oplus 0 =_{E_g} x$.
(XEq2) $x \oplus x =_{E_g} 0$.
(XEq3) $x \oplus y =_{E_g} y \oplus x$.
(XEq4) $x \oplus (y \oplus z) =_{E_g} (x \oplus y) \oplus z$.
(PEq1) $\pi_1(x||y) =_{E_g} x$.
(PEq2) $\pi_2(x||y) =_{E_g} y$.

$||$ is intended to model concatenation, $\oplus$ is the classical XOR and π_j are the projections. Next rules are consequences of the closure rules from Section 4.
(SyE) If $\phi_1 \cong \phi_2$ then $\phi_2 \cong \phi_1$.
(TrE) If $\phi_1 \cong \phi_2$ and $\phi_2 \cong \phi_3$ then $\phi_1 \cong \phi_3$.
(XE1) If $r \notin (fn(\sigma) \cup fn(T))$ then $\nu\widetilde{n}.r.\{\sigma, x = r \oplus T\} \cong \nu\widetilde{n}.r.\{\sigma, x = r\}$.
(CD1) If $(\phi \not\models T_1 \vee \phi \not\models T_2)$ then $\phi \not\models T_1||T_2$.
(HD1) If $\nu\widetilde{n}.\sigma \not\models T$ and $h(T) \notin st(\sigma)$ then $\nu\widetilde{n}.\{\sigma, x = h(T)\} \not\models T$.
(XD1) If $\nu\widetilde{n}.\sigma \not\models T$ and $r \notin (\widetilde{n} \cup fn(T))$ then $\nu\widetilde{n}.r.\{\sigma, x = r \oplus T\} \not\models T$.

5.1 Trapdoor one-way functions in the symbolic model

We extend the above algebra in order to model trapdoor one-way functions. We add a sort $iData$ and new symbols $f : Data \times Data \to iData$,$f^{-1} : iData \times Data \to Data$, $pub : Data \to Data$. f is a trapdoor permutation, with f^{-1} being the inverse function. We extend the equational theory:
(OEq1) $f^{-1}(f(x, pub(y)), y) =_{E_g} x$.

To simplify the notations, we will use $f_k(\bullet)$ instead of $f(\bullet, pub(k))$. Now we want to capture the one wayness of function f. Computationally, a one-way function only ensures the weakly secrecy of a random argument r (as long as the key k is not disclosed to the adversary). Hence we define $S_i = \emptyset$ and $S_d = \{(\nu k.r.\{x_k = pub(k), x = f_k(r)\}, r)\}$.

The following frame encodes the encryption scheme proposed by Bellare and Rogaway in [11]:
$\phi_{br}(m) = \nu k.r.\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus m, z = h(m||r)\}$
where m is the plaintext to be encrypted, f is a trapdoor one-way function, and g and h are hash functions (hence oracles in the ROM model).

$$\textit{TrE}\ \dfrac{\textit{HE1}\ \dfrac{\textit{CD1}\ \dfrac{\textit{GD5}\ \dfrac{\textit{HD1}\ \dfrac{\textit{GD5}\ \dfrac{\textit{OD1}\ \dfrac{}{\{\sigma_2\} \not\models r}}{\{\sigma_2, y = s'\} \not\models r}}{\{\sigma_2, y = g(r)\} \not\models r}}{\{\sigma_2, y = g(r) \oplus p(x_k), z = t\} \not\models r}}{\{\sigma_2, y = g(r) \oplus p(x_k), z = t\} \not\models p(x_k)||r}}{\{\sigma_2, y = g(r) \oplus p(x_k), z = h(p(x_k)||r)\} \cong \{\sigma_2, y = g(r) \oplus p(x_k), z = t\}} \qquad (T1)$$
$$\overline{\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus p(x_k), z = h(p(x_k)||r)\} \cong \{x_k = pub(k), x_a = f_k(r), y = s, z = t\}}$$

Fig. 1. Proof of IND-CPA security of Bellare-Rogaway scheme.

$$\textit{GE1}\ \dfrac{\textit{TrE}\ \dfrac{\textit{GE1}\ \dfrac{\textit{HE1}\ \dfrac{\textit{GD5}\ \dfrac{\textit{OD1}\ \dfrac{}{\{\sigma_2\} \not\models r}}{\{\sigma_2, y = s\} \not\models r}}{\{\sigma_2, y = g(r)\} \cong \{\sigma_2, y = s\}}}{\{\sigma_2, y = g(r) \oplus p(x_k)\} \cong \{\sigma_2, y = s \oplus p(x_k)\}} \textit{XE1}\ \dfrac{}{\{\sigma_2, y = s \oplus p(x_k)\} \cong \{\sigma_2, y = s\}}}{\{\sigma_2, y = g(r) \oplus p(x_k)\} \cong \{\sigma_2, y = s\}}}{\{\sigma_2, y = g(r) \oplus p(x_k), z = t\} \cong \{\sigma_2, y = s, z = t\}}$$

Fig. 2. Tree $(T1)$ from Figure 1.

Now we can see the necessity of p-variables in order to encode IND-CPA security of an encryption scheme. Proving that it holds for any two messages m_1 and m_2

$$\nu k.r.\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus m_1, z = h(m_1||r)\} \cong$$

$$\nu k.r.\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus m_2, z = h(m_2||r)\}$$

is not enough. We did not capture that the adversary is adaptive and she can choose her challenges depending on the public key. Hence we must prove a more stronger equivalence, namely that it holds for any terms $p(x_k)$ and $p'(x_k)$

$$\nu k.r.\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus p(x_k), z = h(p(x_k)||r)\} \cong$$

$$\nu k.r.\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus p'(x_k), z = h(p'(x_k)||r)\}$$

The reader noticed that for asymmetric encryption, this suffices to ensure IND-CPA: possessing the public key and having access to hash-oracles, suffices to encrypt any message, hence it is not necessary to have an oracle to encrypt messages.

Actually, in our case it suffices to prove $\nu k.r.\{x_k = pub(k), x_a = f_k(r), y = g(r) \oplus p(x_k), z = h(p(x_k)||r)\} \cong \nu k.r.s.t.\{x_k = pub(k), x_a = f_k(r), y = s, z = t\}$. By transitivity, this implies: for any two challenges that adversary chooses for $p(x_k)$, the distributions she gets are indistinguishable.

Before proceeding with the proof, we first state some rules that are consequences of the definition of S_d and of the closure rules from Section 4.
(OD1) If f is a one-way function, then $\nu k.r.\{x_k = pub(k), x = f_k(r)\} \not\models r$.
(ODg1) If f is a one-way function and $\nu\widetilde{n}.\nu k.\{x_k = pub(k), x = T\} \cong \nu r.\nu k.\{x_k = pub(k), x = r\}$, then $\nu\widetilde{n}.\nu k.\{x_k = pub(k), x = f_k(T)\} \not\models T$.

The proof of IND-CPA security of Bellare-Rogaway scheme is presented in Figure 1. To simplify the notations we suppose that all names in frames are restricted and we note $\sigma_2 \equiv x_k = pub(k), x_a = f_k(r)$.

5.2 Partially one-way functions in the symbolic model

In this subsection, we show how we can deal with trapdoor partially one-way functions. This extension is motivated by Pointcheval's construction in [23]. In contrast to the previous subsection, we demand for function f a stronger property than one-wayness. Let $Data_1$ be a new sort, and let $f : Data_1 \times Data \times Data \to iData$ be a function and let $f^{-1} : iData \times Data \to Data_1$, such that
(OEq1) $f(f^{-1}(x, y), z, pub(y)) =_{E_g} x$.

The function f is said *partially one way*, if for any given $f(s, r, pub(k))$, it is impossible to compute in polynomial time a corresponding s without the trapdoor k. In order to deal with the fact that f is now partially one-way, we define $S_i = \emptyset$ and $S_d = \{(\nu k.r.s.\{x_k = pub(k), x = f_k(r, s)\}, r)\}$.

The following frame encodes the encryption scheme proposed by Pointcheval in [23].
$\phi_{po}(m) = \nu k.r.s.\{x_k = pub(k), x_a = f_k(r, h(m||s)), y = g(r) \oplus (m||s)\}$
where m is the plaintext to be encrypted, f is a trapdoor partially one-way function, and g and h are hash functions. To prove IND-CPA security of this scheme, we can show in our framework that $\nu k.r.s.\{x_k = pub(k), x_a = f_k(r, h(p(x_k)||s)), y = g(r) \oplus (p(x_k)||s)\} \cong \nu k.r.s_1.s_2.\{x_k = pub(k), x_a = f_k(r, s_1), y = s_2\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\sigma_2, x=r, y=s_2\oplus(p(x_k)||s)\}\cong\{\sigma_2, x=r, y=s_2\}}XE1}{\{\sigma_2, y=s_2, x=r\}\cong\{\sigma_2, y=s_2\oplus(p(x_k)||s), x=r\}}SyE \quad \dfrac{\dfrac{}{\{\sigma_2\}\not\models r}ODp1}{\{\sigma_2, y=s_2\}\not\models r}GD5}{\{\sigma_2, y=s_2\oplus(p(x_k)||s)\}\not\models r}GD6}{\{\sigma_2, y=g(r)\oplus(p(x_k)||s)\}\cong\{\sigma_2, y=s_2\oplus(p(x_k)||s)\}}HE1}{\{x_k=pub(k), x_a=f_k(r, h(p(x_k)||s)), y=g(r)\oplus(p(x_k)||s)\}\cong\{x_k=pub(k), x_a=f_k(r, s_1), y=s_2\}}TrE \quad (T2)$$

Fig. 3. Proof of IND-CPA security of Pointcheval scheme.

$$\dfrac{\dfrac{}{\{\sigma_2, y=s_2\oplus(p(x_k)||s)\}\cong\{\sigma_2, y=s_2\}}XE1 \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset\not\models s}GD1}{\{x_k=pub(k), x_a=f_k(r, s_1)\}\not\models s}GD5}{\{x_k=pub(k), x_a=f_k(r, s_1)\}\not\models p(x_k)||s}CD1}{\{\sigma_2\}\cong\{x_k=pub(k), x_a=f_k(r, s_1)\}}HE1}{\{\sigma_2, y=s_2\}\cong\{x_k=pub(k), x_a=f_k(r, s_1), y=s_2\}}GE1}{\{\sigma_2, y=s_2\oplus(p(x_k)||s)\}\cong\{x_k=pub(k), x_a=f_k(r, s_1), y=s_2\}}TrE$$

Fig. 4. Tree $(T2)$ from Figure 3.

Before proceeding with the proof we first state the next rule that is a consequence of the definition of S_d.
(ODp1) If f is an one-way function, then $\nu k.r.s.\{x_k = pub(k), x = f_k(r, s)\} \not\models r$.

The proof of IND-CPA security of Pointcheval scheme is presented in Figure 3. To simplify notations we suppose that all names in frames are restricted and we note $\sigma_2 \equiv x_k = pub(k), x_a = f_k(r, h(p(x_k)||s))$.

5.3 Computational Diffie Hellman Assumption

In this subsection we prove indistinguishability under chosen plaintext attacks of a variant of Hash-ElGamal encryption scheme ([26]) in the random oracle model under the CDH assumption. The proof of the original scheme([7]) can be easily obtained from our proof and it can be done entirely in our framework.

We will consider two sorts G and A, symbol functions $exp : G \times A \to G$, $* : A \times A \to A$, $0_A : A$, $1_A : A$, $1_G : G$. To simplify the notation we write M^N instead of $exp(M, N)$. We extend the equational theory E_g by the following equations:
(XEqe1) $(x^y)^z =_{E_g} x^{y*z}$.
(XEqe2) $x^{1_A} =_{E_g} x$.
(XEqe3) $x^{0_A} =_{E_g} 1_G$.

To capture the Computational Diffie Hellman Assumption in the symbolic model we define $S_i = \emptyset$ and $S_d = \{(\nu g.r.s.\{x_g = g, x = g^s, y = g^r\}, g^{s*r})\}$.

So we have the next rule that is a consequence of the definition of S_d.
(CDH) $\nu g.r.s.\{x_g = g, x = g^s, y = g^r\} \not\models g^{s*r}$.

The following frame encodes the Hash-ElGamal encryption scheme.
$\phi_{hel}(m) = \nu g.r.s.\{x_g = g, x = g^s, y = g^r, z = h(g^{s*r}) \oplus m\}$
where m is the plaintext to be encrypted, (g, g^s) is the public key and h is a hash function.

The proof of IND-CPA security of Hash-ElGamal's scheme is provided in Figure 5. To simplify the notations we suppose that all names are restricted and we note $\sigma_e \equiv x_g = g, x = g^s, y = g^r$.

$$TrE\ \frac{GE1\ \dfrac{HE1\ \dfrac{GD5\ \dfrac{CDH\ \dfrac{}{\{\sigma_e\} \not\models g^{s*r}}}{\{\sigma_e, z = t\} \not\models g^{s*r}}}{\{\sigma_e, z = h(g^{s*r})\} \cong \{\sigma_e, z = t\}}}{\{\sigma_e, z = h(g^{s*r}) \oplus p(x, x_g)\} \cong \{\sigma_e, z = t \oplus p(x, x_g)\}}\ XE1\ \dfrac{}{\{\sigma_e, z = t \oplus p(x, x_g)\} \cong \{\sigma_e, z = t\}}}{\{x_g = g, x = g^s, y = g^r, z = h(g^{s*r}) \oplus p(x, x_g)\} \cong \{x_g = g, x = g^s, y = g^r, z = t\}}$$

Fig. 5. Proof of IND-CPA security of Hash-ElGamal's scheme

6 Static equivalence and FIR

In this section we adapt the definition of deductibility and static equivalence ([9]) to our framework. After, we justify why they are too coarse to be appropriate abstractions for indistinguishability and weak secrecy. We also prove that in general they are coarser approximations of indistinguishability and weak secrecy than FIR and FNDR.

If ϕ is a frame, and M, N are terms, then we write $(M =_E N)\phi$ for $M\phi =_E N\phi$.

Definition 8 (Deductibility). *A (closed) term T is* **deductible** *from a frame ϕ where $(p_i)_{i\in I} = pvar(\phi)$, written $\phi \vdash T$, if and only if there exists a term M and a set of terms $(M_i)_{i\in I}$, such that $var(M) \subseteq dom(\phi)$,$ar(M_i) = ar(p_i)$, $fn(M, M_i) \cap n(\phi) = \emptyset$ and $(M =_E T)(\phi[(M_i(T_{i_1}, \ldots, T_{i_k})/p_i(T_{i_1}, \ldots, T_{i_k}))_{i\in I}])$. We denote by $\not\vdash$ the logical negation of $\vdash$.*

For instance, we consider the equational theory E_g and the frame $\phi = \nu k_1.k_2.s_1.s_2.\{x_1 = k_1, x_2 = k_2, x_3 = h((s_1 \oplus k_1) \oplus p(x_1, x_2)), x_4 = h((s_2 \oplus k_2) \oplus p(x_1, x_2))\}$. Then $h(s_1) \oplus k_2$ is deductible from ϕ since $h(s_1) \oplus k_2 =_{E_g} x_3[x_1/p(x_1, x_2)] \oplus x_2$ but $h(s_1) \oplus h(s_2)$ is not deductible.

If we consider the frame $\phi' = \nu k.r.s.\{x_k = pub(k), x = f_k(r||s)\}$ where f is a trapdoor one-way function, then neither $r||s$, nor r is deductible from ϕ'. So, the one-wayness of f is modelled by the impossibility of inverting f if k is not disclosed. While this is fair for $r||s$ according to the computational guarantees of f, it seems too strong of assuming that r alone cannot be computed if f is "just" one-way. This raises some doubts about the fairness of $\not\vdash$ as a good abstraction of weak secrecy. We can try to correct this and add an equation of the form $g(f(x||z, pub(y)), y) =_{E_g} x$.

And now, what about r_1, if one gives $f((r_1||r_2)||s)$? In the symbolic setting r_1 is not deductible; in the computational one we have no guarantee; hence, when one stops to add equations? Moreover, in this way we could exclude "good" one-way functions:

in the computational setting, if f is a one-way function, then $f'(x||y) \stackrel{def}{=} x||f(y)$, is another one-way function. The advantage of defining non-deductibility as we did it in the Section 4, is that first, we capture "just" what is supposed to be true in the computational setting, and second, if we add more equations to our abstract algebra (because we discovered that the implementation satisfies more equations) in a coherent manner with respect to the initial

computational assumptions, then our proofs still remain computationally sound. This is not true for $\vdash$.

Definition 9 (Test). *A* **test** *for a frame ϕ is a triplet $((M_i)_{i\in I}, M, N)$ such that $var(M,N) \subseteq dom(\phi)$,$ar(M_i) = ar(p_i)$, $fn(M,N,M_i) \cap n(\phi) = \emptyset$. Then ϕ* **passes the test** *$((M_i)_{i\in I}, M, N)$ if and only if $(M =_E N)(\phi[(M_i(T_{i_1},\ldots,T_{i_k})/p_i(T_{i_1},\ldots,T_{i_k}))_{i\in I}])$.*

Definition 10 (Statically Equivalent). *Two frames ϕ_1 and ϕ_2 are* **statically equivalent***, written as $\phi_1 \approx_E \phi_2$, if and only if*
(i) $dom(\sigma_1) = dom(\sigma_2)$;
(ii) for any test $((M_i)_{i\in I}, M, N)$, ϕ_1 passes the test $((M_i)_{i\in I}, M, N)$ if and only if ϕ_2 passes the test $((M_i)_{i\in I}, M, N)$.

For instance, the two frames $\phi_1 = \nu k.s.\{x_1 = k, x_2 = h(s) \oplus (k \oplus p(x_1))\}$ and $\phi_2 = \nu k.s.\{x_1 = k, x_2 = s \oplus (k \oplus p(x_1))\}$ are statically equivalent with respect to E_g. However the two frames $\phi_1' = \nu k.s.\{x_1 = k, x_2 = h(s) \oplus (k \oplus p(x_1)), x_3 = h(s)\}$ and $\phi_2' = \nu k.s.\{x_1 = k, x_2 = s \oplus (k \oplus p(x_1)), x_3 = h(s)\}$ are not. The frame ϕ_2' passes the test $((x_1), x_2, x_3)$, but ϕ_1' does not.

Let us now consider the equational theory from subsection 5.2. Then the following frames $\nu g.a.b.\{x_1 = g, x_2 = g^a, x_3 = g^b, x_4 = g^{a*b}\}$ and $\nu g.a.b.c.\{x_1 = g, x_2 = g^a, x_3 = g^b, x_4 = g^c\}$ are statically equivalent. This seems right, it is the Decisional Diffie-Hellman assumption. So, a computational implementation that satisfies indistinguishability for the interpretations of this two frames will simply satisfy the DDH assumption. But soundness would imply much more. Even $\nu g.a.b.\{x_1 = g, x_2 = g^a, x_3 = g^b, x_4 = g^{a^2*b^2}\}$ and $\nu g.a.b.c.\{x_1 = g, x_2 = g^a, x_3 = g^b, x_4 = g^c\}$ will be statically equivalent. It is unreasonable to assume that this is true for the computational setting. And as for non-deductibility, the advantage of considering FIR as the abstraction of indistinguishability, is that if we are adding equations in a coherent manner with respect to the initial computational assumptions (that is with S_i), then our proofs still remain computationally sound.

Next proposition says that if the initial sets S_d and S_i are reasonable, then the obtained FIR and FNDR are finer approximations of indistinguishability and weak secrecy than $\vdash$ and $\approx_E$.

Proposition 1. *Let (S_d, S_i) be such that $S_d \subseteq \vdash$ and $S_i \subseteq \approx_E$. Then $\langle S_d\rangle_{\not\vdash} \subseteq \vdash$ and $\langle S_i\rangle_{\cong} \subseteq \approx_E$.*

7 Conclusion

In this paper we developed a general framework for relating formal and computational models for generic encryption schemes in the random oracle model. We proposed general definitions of formal indistinguishability relation and formal non-derivability relation, that is symbolic relations that are computationally sound by construction. We extended previous work with respect to several aspects. First, our framework can cope with adaptive adversaries. This is mandatory in order to prove IND-CPA security. Second, many general constructions use one-way functions, and often they are analyzed in the random oracle model: hence the necessity to capture the weak secrecy in the computational world. Third, the closure rules we propose are designed with the objective of minimizing the initial relations which depend of the cryptographic primitives and assumptions. We illustrated our framework on the generic encryption scheme proposed by Bellare and Rogaway [11] and on Hash El Gamal [7].

As future works, we project to study the (relative) completeness of various equational symbolic theories. Another ambitious extension will be to capture fully active adversaries.

References

1. M. Abadi and P. 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.
2. Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. In Luca Aceto and Anna Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2006.
3. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL*, pages 104–115, 2001.
4. Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. In Chris Hankin, editor, *ESOP*, volume 1381 of *Lecture Notes in Computer Science*, pages 12–26. Springer, 1998.
5. M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable dolev-yao style cryptographic library. In *CSFW*, pages 204–218. IEEE Computer Society, 2004.
6. M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In *ESORICS*, volume 2808 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2003.
7. Joonsang Baek, Byoungcheon Lee, and Kwangjo Kim. Secure length-saving elgamal encryption under the computational diffie-hellman assumption. In Ed Dawson, Andrew Clark, and Colin Boyd, editors, *ACISP*, volume 1841 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2000.
8. Gergei Bana, Payman Mohassel, and Till Stegers. Computational soundness of formal indistinguishability and static equivalence. In Mitsu Okada and Ichiro Satoh, editors, *ASIAN*, volume 4435 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2006.
9. Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer, 2005.
10. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT'04*, volume 950 of *LNCS*, pages 92–111, 1994.
11. Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS'93*, pages 62–73, 1993.
12. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
13. Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.
14. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In Sagiv [24], pages 157–171.
15. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
16. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptol.*, 1(2):77–94, 1988.
17. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.
18. R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In Sagiv [24], pages 172–185.
19. P. Laud. Symmetric encryption in automatic analyses for confidentiality against adaptive adversaries. In *Symposium on Security and Privacy*, pages 71–85, 2004.
20. 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. Springer, 2004.
21. T. Okamoto and D. Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA'01*, pages 159–175, 2001.
22. Duong Hieu Phan and David Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 182–197. Springer, 2004.
23. D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *PKC'00*, pages 129–146, 2000.
24. 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*, 2005.

25. V. Shoup. Oaep reconsidered. *J. Cryptology*, 15(4):223–249, 2002.
26. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. cryptology eprint archive, report 2004/332, 2004.
27. Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *J. on Selected Areas in Communications*, 11(5):715–724, 1993.

A Proofs

A.1 Proof of Theorem 1

Let define $(D_1, I_1) \wedge (D_2, I_2) \stackrel{def}{=} (D_1 \cap D_2, I_1 \cap I_2)$.

Let (S_d, S_i) be some well-formed pair of relations. The existence of the unique smallest (with respect to $\sqsubset$) pair $(\langle S_d \rangle_{\not\models}, \langle S_i \rangle_{\cong})$ is implied by the fact that

1. $(\mathbf{F} \times \mathbf{T}, \mathbf{F} \times \mathbf{F})$ is a $(FNDR, FIR)$ such that $(S_d, S_i) \sqsubset (\mathbf{F} \times \mathbf{T}, \mathbf{F} \times \mathbf{F})$;
2. if (D_1, I_1) and (D_2, I_2) are $(FNDR, FIR)$, then $(D_1, I_1) \wedge (D_2, I_2)$ is a $(FNDR, FIR)$.

Hence, $(\langle S_d \rangle_{\not\models}, \langle S_i \rangle_{\cong})$ can be defined as follows

$$(\langle S_d \rangle_{\not\models}, \langle S_i \rangle_{\cong}) \stackrel{def}{=} \bigwedge \big\{(D, I) | (D, I) \text{ is a } (FNDR, FIR) \text{ such that } (S_d, S_i) \sqsubset (D, I)\big\}.$$

Actually, it is easy to see easy that $(\langle S_d \rangle_{\not\models}, \langle S_i \rangle_{\cong})$ is the least fixed point of some continuous function $\mathcal{F}_{(\not\models, \cong)} : (\mathbf{F} \times \mathbf{T}) \times (\mathbf{F} \times \mathbf{F}) \mapsto (\mathbf{F} \times \mathbf{T}) \times (\mathbf{F} \times \mathbf{F})$ defined following the rules *(GD1)*, ..., *(GD6)*,*(GE1)*,...,*(GE4)*, *(HE1)*, symmetry and transitivity. It can be constructed by applying iteratively $(\langle S_d \rangle_n, \langle S_i \rangle_n) = \mathcal{F}^n_{(\not\models, \cong)}((S_d, S_i))$, with $n \in \mathbb{N}$ until reaching a fixpoint.

Now we prove that $\langle S_d \rangle_{\not\models}$ is $\not\vdash$-sound and $\langle S_i \rangle_{\cong}$ is $\approx$-sound, provided that $=_E$ is $=$-sound, S_d is $\not\vdash$-sound and S_i is $\approx$-sound.

Most of the closure rules have premises that assume some hypothesis on $\not\models$ or $\cong$. Let suppose that for any such closure rule (R), we prove its computational soundness, that is, the following fact:

Fact A1 *For any adversary $\mathcal{A}$ against the conclusion of the rule (R), there exists some adversary $\mathcal{B}$ (or tuple of adversaries $\mathcal{B}_i$) breaking one of the premises of (R) , and moreover:*

1. *the advantage of $\mathcal{A}$ is a polynomial w.r.t. to η and the advantage of $\mathcal{B}$ (advantages of $\mathcal{B}_i$, respectively) and*
2. *the adversary $\mathcal{A}$ has an execution time which is a polynomial w.r.t. to η and the execution time of $\mathcal{B}$ (execution times of $\mathcal{B}_i$, respectively).*

Now let suppose that there is some element (e_1, e_2) in $\langle S_d \rangle_{\not\models}$ or $\langle S_i \rangle_{\cong}$ which is not $\not\vdash$-sound or $\approx$-sound. Let n be the number of steps needed to include (e_1, e_2) in $\langle S_d \rangle_{\not\models}$ or $\langle S_i \rangle_{\cong}$, i.e. the minimal number of iterations $(\langle S_d \rangle_n, \langle S_i \rangle_n)$ needed to get $(e_1, e_2) \in \langle S_d \rangle_n$ or $(e_1, e_2) \in \langle S_i \rangle_n$.

Then, for any adversary $\mathcal{A}_0$ against the soundness of (e_1, e_2), we can construct an adversary $\mathcal{A}_n$ against the soundness of an element (e_1^0, e_2^0) of S_d or S_i, such that

1. the advantage of $\mathcal{A}_0$ is bounded by an expression which depends of n and which is a polynomial w.r.t. η and the advantage of $\mathcal{A}_n$, and
2. the execution time of $\mathcal{A}_n$ is bounded by an expression which depends of n and which is a polynomial w.r.t. η and the execution time of $\mathcal{A}_0$.

Since our reasoning is asymptotically (and n is independent from η), this would imply that (e_1^0, e_2^0) is not sound, contradiction with the $\not\vdash$-soundness of S_d or the $\approx$-soundness of S_i.

In what follows we prove soundness for all rules of section 4.

(GD1) $\nu r.\emptyset \not\models r$.

Proof. To easy notations, we note $S = [\![s]\!]$. Then we have

$$\begin{aligned}\mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \mathcal{A}() = bs] &= \sum_{bs \in S} \mathsf{Pr}[bs' \xleftarrow{r} [\![s]\!] : bs' = bs] * \mathsf{Pr}[\mathcal{A}() = bs] \\ &= \sum_{bs \in S} \frac{1}{|S|} * \mathsf{Pr}[\mathcal{A}() = bs] = \frac{1}{|S|} * \sum_{bs \in S} \mathsf{Pr}[\mathcal{A}() = bs] = \frac{1}{|S|}\end{aligned}$$

Now we use the assumption that all sorts are suppposed to be of size exponential in η. □

(GD2) If $\phi \not\models M$ then $\tau(\phi) \not\models \tau(M)$ for any renaming τ.

Proof. Using the fact that renamings do not change distributions, we get $[\![\tau(\phi), \tau(M)]\!] = [\![\phi, M]\!]$. □

(GD3) If $\phi \not\models M$ then $\phi \not\models N$ for any term $N =_E M$.
(GD4) If $\phi \not\models M$ then $\phi' \not\models M$ for any frame $\phi' =_E \phi$.

Proof. Obviously, using the $=$-soundness of $=_E$. □

(GD5) If $\phi \not\models M$ then $\phi'\phi \not\models M$ for any frame ϕ' such that $var(\phi') \subseteq dom(\phi)$ and $n(\phi') \cap bn(\phi) = \emptyset$.

Proof. Let ϕ' such that $var(\phi') \subseteq dom(\phi)$ and $n(\phi') \cap bn(\phi) = \emptyset$. Let us suppose that $\phi \not\models M$ is $\not\sim$-sound, and let us prove that $\phi'\phi \not\models M$ is also $\not\sim$-sound. We have to show that for any probabilistic polynomial-time adversary $\mathcal{A}$ against $\phi'\phi \not\models M$, there exists an adversary $\mathcal{B}$ against $\phi \not\models M$ that satisfies the conditions of Fact A1.

The adversary $\mathcal{B}$ uses $\mathcal{A}$ as a black box to first compute $\hat{\phi} \xleftarrow{r} [\![\phi]\!]_{\mathcal{A}}$; then it interprets all variables in $var(\phi')$ by bitstrings obtained in the previous stage (as $var(\phi') \subseteq dom(\phi)$); it continues to use $\mathcal{A}$ as a black box in order to interpret all queries from $pvar(\phi')$; finally it gets a concrete frame from $[\![\phi'\phi]\!]_{\mathcal{A}}$ and passes it to $\mathcal{A}$; it answers as $\mathcal{A}$. Hence, the advantage of $\mathcal{B}$ equals the advantage of $\mathcal{A}$, $\mathsf{Adv}(\mathcal{B}, \eta, \phi \not\models M) = \mathsf{Pr}[\hat{\phi}, \hat{e} \xleftarrow{r} [\![\phi, M]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] = \mathsf{Pr}[\hat{\phi''}, \hat{e} \xleftarrow{r} [\![\phi'\phi, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi''}) = \hat{e}] = \mathsf{Adv}(\mathcal{A}, \eta, \phi'\phi \not\models M)$.

In addition, the execution time of $\mathcal{B}$ is a polynomial w.r.t. to η and the execution time of $\mathcal{A}$, using that the size of encoding of ϕ' is constant in η.

□

(GD6) If T, U are terms such that $U[T/y] =_E z$ and $z \in var(T) \setminus var(U)$ and $(fn(T) \cup fn(U)) \cap \widetilde{n} = \emptyset$, then for all substitutions σ_1, σ_2 such that $x \notin dom(\sigma_i)$ and $\nu\widetilde{n}.\{\sigma_1, x = T[M/z]\} \cong \nu\widetilde{n}.\{\sigma_2, x = T[N/z]\}$ and $\nu\widetilde{n}.\sigma_1 \not\models M$ then $\nu\widetilde{n}.\sigma_2 \not\models N$.

Proof. Let us suppose that $\nu\widetilde{n}.\sigma_1 \not\models M$ is $\not\sim$-sound and $\nu\widetilde{n}.\{\sigma_1, x = T[M/z]\} \cong \nu\widetilde{n}.\{\sigma_2, x = T[N/z]\}$ is $\approx$-sound, and let us prove that $\nu\widetilde{n}.\sigma_2 \not\models N$ is also $\not\sim$-sound.

We have to show that for any probabilistic polynomial-time adversary $\mathcal{A}$ against $\nu\widetilde{n}.\sigma_2 \not\models N$, there exists adversaries $\mathcal{B}_1$ against $\nu\widetilde{n}.\{\sigma_1, x = T[M/z]\} \cong \nu\widetilde{n}.\{\sigma_2, x = T[N/z]\}$, and $\mathcal{B}_2$ against $\nu\widetilde{n}.\sigma_1 \not\models M$ which satisfy the conditions of Fact A1.

In our case we will provide an adversary $\mathcal{B}$ to play the role of $\mathcal{B}_1$ and we will use the adversary $\mathcal{A}$ as player for the role of $\mathcal{B}_2$, too.

Since $fn(T) \cap \widetilde{n} = \emptyset$, it follows that $T(z)$ is constructible using only $dom(\sigma_i)$. Hence, the adversary $\mathcal{B}$ uses $\mathcal{A}$ as a black box to first get either $(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\{\sigma_1, x = T[M/z]\}]\!]_{\mathcal{A}}$ or $(\hat{\phi}, x = \hat{t}(\hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\{\sigma_2, x = T[N/z]\}]\!]_{\mathcal{A}}$. Then $\mathcal{A}$ stops and answers some string bs. If $\hat{t}(\hat{e}) = \hat{t}(bs)$, $\mathcal{B}$ answers 1 and stops. If $\hat{t}(\hat{e}) \neq \hat{t}(bs)$, $\mathcal{B}$ picks randomly a bit c, answers c and stops. From the definition of $\mathcal{B}$, and using the $=_E$ injectivity of T and the $=$-soundness of $=_E$, we have the following:

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, x = T[N/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] = 1,$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, x = T[M/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] = 1,$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, x = T[N/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] = \frac{1}{2} + n_2(\eta)$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, x = T[M/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] = \frac{1}{2} + n_1(\eta)$

where $n_1(\eta)$ and $n_2(\eta)$ are some negligible functions.

Now we have

$\mathsf{Adv}(\mathcal{B}, \eta, \nu\widetilde{n}.\{\sigma_1, x = T[M/z]\}, \nu\widetilde{n}.\{\sigma_2, x = T[N/z]\}) =$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, x = T[N/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1] -$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, x = T[M/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 0] =$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, x = T[N/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] * \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] +$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, x = T[N/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] * \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] -$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, x = T[M/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] * \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] -$

$\mathsf{Pr}[(\hat{\phi}, x = \hat{t}(\hat{e})) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, x = T[M/z]]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi}, x = \hat{t}(\hat{e})) = 1 | \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] * \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] =$

$\mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] + (\frac{1}{2} + n_2(\eta)) * \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] -$

$\mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] - (\frac{1}{2} + n_1(\eta)) * \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) \neq \hat{e}] =$

$\mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] + \frac{1}{2} * (1 - \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}]) -$

$\mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] - \frac{1}{2} * (1 - \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] + n_3(\eta) =$

$\frac{1}{2} * (\mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_2, N]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}] - \mathsf{Pr}[(\hat{\phi}, \hat{e}) \stackrel{r}{\leftarrow} [\![\nu\widetilde{n}.\sigma_1, M]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = \hat{e}])) + n_3(\eta) =$

$\frac{1}{2} * (\mathsf{Adv}(\mathcal{A}, \eta, \nu\widetilde{n}.\sigma \not\models N) - \mathsf{Adv}(\mathcal{A}, \eta, \nu\widetilde{n}.\sigma \not\models M))) + n_3(\eta)$

for some well-chosen negligible function $n_3(\eta)$.

Moreover, it is easy to see that the execution time of $\mathcal{B}$ is a polynomial w.r.t. to η and the execution time of $\mathcal{A}$, using that the test $\hat{t}(\hat{e}) \stackrel{?}{=} \hat{t}(bs)$, and picking uniformly a random bit can be done in a time polynomial w.r.t. to η. □

(GE1) If $\phi_1 \cong \phi_2$ then $\phi\phi_1 \cong \phi\phi_2$, for any frame ϕ such that $var(\phi) \subseteq dom(\phi_i)$ and $n(\phi) \cap bn(\phi_i) = \phi$.

Proof. Let ϕ such that $var(\phi) \subseteq dom(\phi_i)$ and $n(\phi) \cap bn(\phi_i) = \phi$. Let us suppose that $\phi_1 \cong \phi_2$ is $\approx$-sound, and let us prove that $\phi\phi_1 \cong \phi\phi_2$ is also $\approx$-sound. We have to show that for any probabilistic polynomial-time adversary $\mathcal{B}$, $([\![\phi\phi_1]\!]_{\mathcal{B}}) \approx ([\![\phi\phi_2]\!]_{\mathcal{B}})$.

Let us suppose that there exists a probabilistic polynomial-time adversary $\mathcal{B}$ such that $([\![\phi\phi_1]\!]_{\mathcal{B}}) \not\approx ([\![\phi\phi_1]\!]_{\mathcal{B}})$, that is $\mathsf{Pr}[\hat{\phi'} \xleftarrow{r} [\![\phi\phi_1]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi'}) = 1] - \mathsf{Pr}[\hat{\phi'} \xleftarrow{r} [\![\phi\phi_2]\!]_{\mathcal{B}} : \mathcal{B}^{\mathcal{O}}(\hat{\phi'}) = 1]$

is non-negligible.

Then we construct an adversary $\mathcal{A}$ such that $\mathsf{Pr}[\hat{\phi} \xleftarrow{r} [\![\phi_1]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = 1] - \mathsf{Pr}[\hat{\phi} \xleftarrow{r} [\![\phi_2]\!]_{\mathcal{A}} : \mathcal{A}^{\mathcal{O}}(\hat{\phi}) = 1]$ is non-negligible.

The adversary $\mathcal{A}$ uses $\mathcal{B}$ as a black box to first get $\hat{\phi} \xleftarrow{r} [\![\phi_i]\!]_{\mathcal{B}}$; then it interprets all variables in $var(\phi)$ by bitstrings obtained in the previous stage (as $var(\phi) \subseteq dom(\phi_i)$); then it continues to use $\mathcal{B}$ as a black box in order to interpret all queries from $pvar(\phi)$; finally it get a concrete frame from $[\![\phi\phi_i]\!]_{\mathcal{B}}$ and passes it to $\mathcal{B}$; it answers as $\mathcal{B}$. Hence, the advantage of $\mathcal{A}$ equals the advantage of $\mathcal{B}$, which is non-negligible. In addition, $\mathcal{A}$ runs in probabilistic polynomial-time since $\mathcal{B}$ runs in probabilistic polynomial-time and the size of encoding of ϕ is constant in η. This is a contradiction with $\phi_1 \cong \phi_2$ being $\approx$-sound. Hence $\phi\phi_1 \cong \phi\phi_2$ is also $\approx$-sound. □

(GE2) $\phi \cong \phi'$ for any frame ϕ' such that $\phi' =_E \phi$.

Proof. Obviously, using the $=$-soundness of $=_E$. □

(GE3) $\tau(\phi) \cong \phi$ for any renaming τ.

Proof. Using the fact that renamings do not change distributions, we get $[\![\tau(\phi)]\!] = [\![\phi]\!]$. □

(GE4) If M, N are terms of the same sort such that $N[M/z] =_E y$ and $y \in var(M) \setminus var(N)$, then for any substitution σ such that $r \notin (fn(\sigma) \cup fn(M) \cup fn(N))$ and $x \notin dom(\sigma)$ it holds $\nu\widetilde{n}.r.\{\sigma, x = M[r/y]\}) \cong \nu\widetilde{n}.r.\{\sigma, x = r\}$.

Proof. We prove that the statistical distance $d([bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs)], [bs \xleftarrow{r} [\![s]\!] : bs])$ is negligible for any computational functions $\hat{g} : [\![s]\!] \to [\![s]\!]$ and $\widehat{g^{-1}} : [\![s]\!] \to [\![s]\!]$ such that $\mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \widehat{g^{-1}}(\hat{g}(bs)) \neq bs]$ is negligible. Then, the correctness of rule *(GE4)* is easy to prove using the $=$-soundness of $=_E$ and noticing that the context N can be used to build the inverse function of $\lambda r.M(r)$.

Let us suppose that $\mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \widehat{g^{-1}}(\hat{g}(bs)) \neq bs]$ is negligible. To easy notations, we note $S = [\![s]\!]$, $S_1 = \{bs \in S | \widehat{g^{-1}}(\hat{g}(bs)) = bs\}$, $S_2 = \{bs \in S | \widehat{g^{-1}}(\hat{g}(bs)) \neq bs\}$, $s = |S|$, $s_i = |S_i|$. Our hypothesis is equivalent to $s_2 = s * \eta$ for some negligible function η. Also, it easy to see that $\hat{g} : S_1 \to \hat{g}(S_1)$ is an injective function, and hence a bijective function too. So, if $bs' \in \hat{g}(S_1)$ we know that there is exactly one element in S_1 noted $i(bs')$ such that $\hat{g}(i(bs')) = bs'$. We note in this case $S_{1,bs'} = S_1 \setminus \{i(bs')\}$. Moreover, $|S \setminus \hat{g}(S_1)| = |S \setminus S_1| = s_2$.

$$
\begin{aligned}
&d([bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs)], [bs \xleftarrow{r} [\![s]\!] : bs]) \\
&= \sum_{bs' \in S} \left| \mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs) = bs'] - \frac{1}{s} \right| \\
&= \sum_{bs' \in S_1} \left| \mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs) = bs'] - \frac{1}{s} \right| + \sum_{bs' \in S_2} \left| \mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs) = bs'] - \frac{1}{s} \right| \\
&\leq \sum_{bs' \in S_1 \cap \hat{g}(S_1)} \left| \mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs) = bs'] - \frac{1}{s} \right| + \sum_{bs' \in S_1 \cap (S \setminus \hat{g}(S_1))} \left| \mathsf{Pr}[bs \xleftarrow{r} [\![s]\!] : \hat{g}(bs) = bs'] - \frac{1}{s} \right| + \eta \\
&\leq \sum_{bs' \in S_1 \cap \hat{g}(S_1)} \left| \frac{1}{s} * \sum_{bs \in S} \chi_{[\hat{g}(bs) = bs']} - \frac{1}{s} \right| + \eta + \eta \\
&= \sum_{bs' \in S_1 \cap \hat{g}(S_1)} \frac{1}{s} * \left| \chi_{[\hat{g}(i(bs'))) = bs']} + \sum_{bs \in S_{1,bs'}} \chi_{[\hat{g}(bs) = bs']} + \sum_{bs \in S_2} \chi_{[\hat{g}(bs) = bs']} - 1 \right| + 2 * \eta \\
&\leq \sum_{bs' \in S_1 \cap \hat{g}(S_1)} \frac{1}{s} * \left| 1 + 0 + s_2 - 1 \right| + 2 * \eta \\
&= 3 * \eta
\end{aligned}
$$

□

(HE1) If $\nu\widetilde{n}.r.\sigma[r/h(T)] \not\models T$ and $r \notin n(\sigma)$, then $\nu\widetilde{n}.\sigma \cong \nu\widetilde{n}.r.\sigma[r/h(T)]$.

Proof. In the random oracle model, hash functions are drawn uniformly at random from the space of functions of suitable type at the beginning of the interpretation of the frame. Thus, the images that the hash function associates to different inputs are completely independent. Therefore, one can delay the draw of each hash value until needed. We use $\sigma[\bullet]$ for $\sigma[\bullet/h(T)]$, i.e. σ where all occurrences of $h(T)$ are replaced by $\bullet$.

Now, using that $\nu\widetilde{n}.r.\sigma[r/h(T)] \not\models T$ we get

$$
\begin{aligned}
&[\![\nu\widetilde{n}.\sigma[h(T)/\bullet]\}]\!]_{\mathcal{A}} \\
&\quad = \left[\mathcal{H} \xleftarrow{r} \Omega; \mathcal{O} = \mathcal{H}; (\hat{\phi}[\bullet], bs) \xleftarrow{r} [\![(\nu\widetilde{n}.\sigma[\bullet], T)]\!]_{\mathcal{H},\mathcal{A}} : \hat{\phi}[H(bs)/\bullet]\right] \\
&\qquad \text{(since } \nu\widetilde{n}.r.\sigma[r/h(T)] \not\models T \text{ one can delay the draw of } h([\![T]\!])\text{)} \\
&\quad \sim \left[\mathcal{H} \xleftarrow{r} \Omega; \mathcal{O} = \mathcal{H}; (\hat{\phi}[\bullet], bs) \xleftarrow{r} [\![(\nu\widetilde{n}.\sigma[\bullet], T)]\!]_{\mathcal{H},\mathcal{A}}; v \xleftarrow{r} [\![\mathbf{s}(T)]\!]; \right. \\
&\qquad\qquad \left. \mathcal{O} = \mathcal{H}[H \to H[bs \to v]] : (\hat{\phi}[v/\bullet])\right] \\
&\qquad \text{(since } \nu\widetilde{n}.r.\sigma[r/h(T)] \not\models T \text{ the probability that } \mathcal{A} \text{ query } h([\![T]\!]) \text{ is negligible)} \\
&\quad \sim \left[\mathcal{H} \xleftarrow{r} \Omega; \mathcal{O} = \mathcal{H}; (\hat{\phi}[\bullet], bs) \xleftarrow{r} [\![(\nu\widetilde{n}.\sigma[\bullet], T)]\!]_{\mathcal{H},\mathcal{A}}; v \xleftarrow{r} [\![\mathbf{s}(T)]\!] : (\hat{\phi}[v/\bullet])\right] \\
&\quad = \left[\mathcal{H} \xleftarrow{r} \Omega; \mathcal{O} = \mathcal{H}; \hat{\phi}[\bullet] \xleftarrow{r} [\![\nu\widetilde{n}.\sigma[\bullet]]\!]_{\mathcal{H},\mathcal{A}}; v \xleftarrow{r} [\![\mathbf{s}(T)]\!] : (\hat{\phi}[v/\bullet])\right] \\
&\quad = [\![\nu\widetilde{n}.\nu r.\{\sigma[r/\bullet]\}]\!]_{\mathcal{A}}
\end{aligned}
$$

□

(SyE) If $\phi_1 \cong \phi_2$ then $\phi_2 \cong \phi_1$.
(TrE) If $\phi_1 \cong \phi_2$ and $\phi_2 \cong \phi_3$ then $\phi_1 \cong \phi_3$.

Proof. Obviously, using that indistinguishability is an equivalence relation. □

A.2 Proofs of Derived rules from Section 5

(XE1) If $r \notin (fn(\sigma) \cup fn(T))$ then $\nu\widetilde{n}.r.\{\sigma, x = r \oplus T\} \cong \nu\widetilde{n}.r.\{\sigma, x = r\}$.

Proof. Consequence of rule *(GE4)* for $M = y \oplus T$ and $N = z \oplus T$ and equations *(XEqi)*. □

(CD1) If $(\phi \not\models T_1 \vee \phi \not\models T_2)$ then $\phi \not\models T_1||T_2$.

Proof. Consequence of rules *(GD5)* and *(GD3)* and equations *(PEq1)* and *(PEq2)*. □

(HD1) If $\nu\widetilde{n}.\sigma \not\models T$ and $h(T) \notin st(\nu\widetilde{n}.\sigma)$ then $\nu\widetilde{n}.\{\sigma, x = h(T)\} \not\models T$.

Proof. Consequence of rules *(HE1)*, *(GD6)* and *(GD5)*. □

(XD1) If $\nu\widetilde{n}.\sigma \not\models T$ and $r \notin (\widetilde{n} \cup fn(T))$ then $\nu\widetilde{n}.r.\{\sigma, x = r \oplus T\} \not\models T$.

Proof. Consequence of rules *(GD5)*, *(GD6)*, *(HE1)* and *(SyE)*. □

(ODg1) If f is a one-way function and $\nu\widetilde{n}.\{x_k = pub(k), x = T\} \cong \nu r.\{x_k = pub(k), x = r\}$, then $\nu\widetilde{n}.\{x_k = pub(k), x = f_k(T)\} \not\models T$.

Proof. Consequence of rules *(OD1)*, *(GE1)* and *(GD6)*. □