Computational Soundness of Formal Encryption in Coq

Ricardo Corin
MSR-INRIA Joint Centre & University of Twente

Abstract

We formalize Abadi and Rogaway's computational soundness result in the Coq interactive theorem prover. This requires to model notions of provable cryptography like indistinguishability between ensembles of probability distributions, PPT reductions, and security notions for encryption schemes. Our formalization is the first computational soundness result to be mechanized, and it shows the feasibility of rigorous reasoning of computational cryptography inside a generic interactive theorem prover.

1 Introduction

Usually, it is necessary to adopt an abstract view of cryptographic operations to make the design and analysis of cryptographic protocols more manageable. Two different, but still related abstract views of cryptographic operations –the formal and the computational– have developed separately in the last years. In the former, the exchanged messages of the protocol are modelled as formal expressions of a term algebra. The (cryptographic) operations, such as message pairing and encryption, are modelled as term constructors. In this setting, an adversary and its abilities can be modelled in terms of the messages the adversary knows. On the other hand, in the computational model, messages are considered to be (more realistically) bit-strings, while cryptographic operations are seen as functions over these bit-strings. Here, an adversary is modelled as any efficient algorithm.

Both of the two above models have advantages and disadvantages. On the one hand, the formal model allows to reason about cryptographic protocols more easily and generally. However, such benefits arise from the adoption of fairly strong assumptions (such as freeness of the term algebra, and fixing the adversary model). On the other hand, the computational model, by considering messages as bit-strings and modelling the adversary as any efficient algorithm, provides a more realistic model and thus offers more convincing security guarantees. However, proving protocols correct in the computational model is more difficult and less general than in the formal model.

In the seminal work of Abadi and Rogaway [4], it is shown that if two formal expressions are similar to a formal adversary, then their corresponding computational interpretations, represented as bit-strings in the computational model, are also indistinguishable to any computational adversary (*computational soundness*). This result comprises a very important (first) step into relating the formal and computational model, and was followed by several others [3, 6, 9, 10, 12, 11].

Computational soundness theorems are of great value because they simplify the analysis of a given protocol, since we only need to fulfil the formal requirements (which can be handled automatically in several cases) to obtain strong computational results. Moreover, computational soundness results can be proved once and for all, independently of the actual protocol

under analysis. However, this also means that soundness results are critical, and need to be not only correct but also clear and precise, so that we can have full understanding and confidence on our protocol analysis. Hence, soundness results are particularly attractive for rigorous formalization in a prover. In this paper we provide such a formalization for Abadi and Rogaway's computational soundness result in the Coq interactive theorem prover [2]. Our choice of proof assistant is based on the fact that Coq is both quite suitable for (1) manipulating (inductive) datatypes for terms and (fixpoint) functions that compute on them and (2) defining precisely the computational model, including PPT reductions, indistinguishability, and security notions for encryption schemes.

Related Work. Work on formalization in interactive theorem provers started in the symbolic setting by Paulson [13]. Tarento et al. formalized in Coq models for the Generic and Random Oracle models [5] (but did not focus on PPT reduction-based proofs). Recently, Sprenger et al. developed a BPW model in Isabelle/HOL [14] (although this work does not formalize BRSIM/UC cryptographic soundness).

In recent work [8], we considered a Probabilistic Hoare-style logic that can be used to describe game-based cryptographic proofs, and illustrated our technique by proving semantic security of ElGamal. The work presented in this paper can be seen as preparing the ground for formalizing this kind of approaches, by exploring the formalization of core concepts of provable cryptography inside a proof assistant like Coq.

In contrast to applying soundness results from formal to computational settings, Blanchet has recently proposed a protocol verifier called CryptoVerif [7] that directly works in a computational setting. In contrast to soundness settings where the security of operations is "built-in" (e.g., type-0 for symmetric encryption in this paper), CryptoVerif is more flexible in that one can fine-tune the security notion that a given cryptographic operation has. Of course, this extra control may not be always needed, and it does not come for free as CryptoVerif may in some cases require manual guidance.

The paper is structured as follows. The formal view is presented in the next section. Section 3 presents the computational model. Section 4 describes the soundness proof. We conclude in Section 5, with some conclusions learnt by doing this proof and some hopes for the future. A more detailed version of this paper, including Coq proofs, is available [1].

2 Formal View

In the following we illustrate our formalization using (straightforward) excerpts of Coq code. We start with the definition of formal expressions, i.e. terms that are defined using the following inductive datatype.

```
Inductive term : Type :=
  | Bit : ℕ → term
  | Key : ℕ → term
  | Pair : term → term → term
  | Enc : ℕ → term → term
  | Un : term.
```

Bits and keys use natural numbers as labels. We include the undecryptable $\square$, here called Un. For example, terms m_0 = Pair (Key 1) (Enc 1 (Bit 1)) and m_1 = Pair (Key 1) (Enc 2 (Bit 1)) represent the concatenation of key K_1 with the encryption of bit 1 under key K_1 and K_2, respectively.

We define functions for obtaining the recoverable, hidden keys, and entailment, as usual. Following [4], the p function inputs a term m and a sequence of keys ks and computes the pattern, presented below. We also define the equivalences $\equiv$ and $\approx$ (called sequiv and equiv).

```
Fixpoint p (m:term) (ks: seq ℕ) : term :=
  match m with
    | Un ⇒ Un
    | Bit b ⇒ Bit b
    | Key k ⇒ Key k
    | Pair t1 t2 ⇒ Pair (p t1 ks) (p t2 ks)
    | Enc k1 t1 ⇒ if mem ks k1 then Enc k1 (p t1 ks) else Un
  end.
Definition pattern (m: term) :term := p m (recoverable m).
Definition sequiv (m0 m1 : term) : Prop := pattern m0 = pattern m1 .
Definition equiv (m0 m1 : term) : Prop := ∃ σ, perm σ ∧ sequiv m0 (subst m1 σ ).
```

(Here, predicate mem ks k1 tests membership of k1 in ks, perm σ holds iff σ is a permutation on keys, and subst applies a substitution to a term.) For example, for m_0 and m_1 above we have pattern m_0 = m_0 and pattern m_1 = (Key 1, Un), and hence we can prove they are not equivalent. In the soundness proof, we assume given a permutation renaming : term → term that reindexes keys s.t. keys with greater indexes always encrypt keys with smaller indexes (in [4] this is the 'prime' renaming reindexing from M, N to M', N'). We also assume given a function ki: term → ℕ → ℕ which returns the i-th indexed key in renaming m, and a function ks: term → ℕ → (seq ℕ) which returns the sequence of keys indexed from 1 to $i - 1$.

3 Computational Model

In order to carry out the soundness proof, we first formalize some notions of provable cryptography.

Basic notions Randomness is modelled by using *coins*, elements of a finite type $\mathcal{C}$[1]. Each coin is in turn an infinite stream that can be shifted to a new position (thus obtaining a new coin, denoted c_n for shifting c to position n) or projected (thus obtaining randomness that algorithms can use). We assume fixed an ensemble over coins (i.e., a family of probability distributions) $\{Pr_\eta\}$, indexed by security parameters $\eta \in \mathbb{N}$, from coins to the $[0, 1]$ real interval. Using this ensemble, we define in Coq a function that calculates the probability of a predicate P : $\mathcal{C}$ → bool, denoted as $Pr_\eta[P]$, which sums the probabilities $Pr_\eta(c)$ for each coin c s.t. $P\ c$ holds.

Symmetric Encryption A scheme sch consists of a triple of algorithms G, E, D, for key generation (of type ℕ → $\mathcal{C}$ → $\mathcal{BS}$, where $\mathcal{BS}$ are finite sequences of bits), encryption ($\mathcal{C}$ → $\mathcal{BS}$ → $\mathcal{BS}$ → $\mathcal{BS}$), and decryption ($\mathcal{BS}$ → $\mathcal{BS}$ → $\mathcal{BS}$). In addition, we model a number of functional properties (e.g., that decryption inverses encryption), although they are not required by the soundness proof.

Adversaries An adversary A is a Coq function of type TA := $\mathcal{C}$ → $\mathcal{BS}$ → $\mathcal{BS}$. We assume given a predicate testing whether an adversary is PPT, called PPT : TA → bool.

PPT reductions Reductions in which an adversary A' is built out of another A are common in provable cryptography, where it is usually shown that if A is PPT then A' is also PPT. We rigorously model this as follows. Given a *seed* consisting of an adversary A, a scheme sch, and oracles o_1, o_2 (all assumed to be PPT), we build a family of *extended* adversaries $F(A, sch, o_1, o_2)$, i.e. functions that compute on their own and may call A, the oracles o_1, o_2 or the encryption scheme sch at some point. Extended adversaries

[1] The size of $\mathcal{C}$ can be made to depend on the security parameter η (e.g., polynomially large in η).

can be built out of "simpler" extended adversaries by using conditionals, sequential composition, and term recursion (but not unbounded computations like using a 'while' command). Crucially, given a Coq proof of $A' \in F(A, sch, o_1, o_2)$, we can easily measure the number of execution steps done by A' (using a simple function $countSteps$) *before* it calls A or o_i; we can then bound the steps of A' by a polynomial in the security parameter, establishing the reduction to be PPT[2].

Indistinguishability & Type-0 We define negligibility of functions as usual. Let $zero$ denote the empty bitstring. Given an ensemble over coins $\{Pr_\eta\}$, an adversary A, and a family of functions $\{D_\eta\}$ from coins to bitstrings, we let function $\Pr_\eta[A(c_n, D_\eta(c)) == zero]$ denote the resulting probability of A "recognizing" distribution D_η (by outputting $zero$) for security parameter η, starting at c_n; for each c s.t. $A(c_n, D_\eta(c))$ returns $zero$, we add the probability $Pr_\eta(c)$. Indistinguishability between two distributions D_η and D'_η, given by predicate indistinguishable D D', holds when the difference function

$$\Pr_\eta[A(c_n, D_\eta(c)) \ = \ zero] - \Pr_\eta[A(c_n, D'_\eta(c)) \ = \ zero]$$

is a negligible function in η for every A s.t. PPT η A. Here, $n \geq 0$ is a 'shift' parameter telling in which position can A start using its randomness (i.e., at c_n, but not before); this allows D to use private randomness (starting in c_0 until c_{n-1}) which is hidden from A. In the soundness proof, for terms M and N we let $D = [\![M]\!]$, $D' = [\![N]\!]$ and n is the number of encryptions and keys in M and N (given by function shf N M).

Type-0 security is defined analogously, although now no input is directly given to the attacker (in contrast to indistinguishability above in which A is given $D_\eta(c)$). Rather, oracles get instantiated differently in the left and right instances, and the role of A is now to guess to which oracles it is speaking to. Given two extended adversaries A_0, A'_0, we define a Coq predicate "same up to oracles" which takes two proofs of $A_0 \in F(A, sch, o_1, o_2)$ and $A'_0 \in F(A, sch, o'_1, o'_2)$ that holds iff A_0 is the exactly the same adversary as A'_0, except that when A_0 calls o_i, A'_0 calls o'_i. We can now define type-0 security of a scheme sch of algorithms G, E, D, given by predicate type0scheme sch, which holds when the difference function

$$\Pr_\eta[A_0(c_n, zero) = zero] - \Pr_\eta[A'_0(c_n, zero) = zero]$$

is negligible in η for every A_0 and A'_0 same up to oracles, extensions of A using "real" and "fake" oracles, respectively (formally, $A_0 \in F(A, sch, E(G_\eta(c_0), \cdot), E_\eta(G_\eta(c_1), \cdot))$ and $A'_0 \in F(A, sch, E(G_\eta(c_0), zero), E(G_\eta(c_0), zero)))$, where G and E are the key generation and encryption functions of scheme sch.

Note here that c_0 is used to generate the left key of the oracles, and c_1 is the right key of only the left "real" oracle. Adversaries A_0 and A'_0 are given c_n (for $n \geq 2$) to avoid the attacker from knowing c_0 and c_1 trivially. Type-1, 2, and 3 can be defined analogously.

These notions are in fact generic in provable cryptography, and could be used in other developments; they are not specific to the soundness proof, which we elaborate next.

4 Soundness

We define a function from terms to bitstrings called convert. A call to convert index m, denoted [[m]] index, returns a mapping that given a security parameter η and a coin c converts term m to a bitstring, using the encryption scheme sch instantiated by the security parameter η, starting from c_{index}. We are now ready to state the soundness theorem:

[2] In fact, since (the number of instructions of) extended adversaries do not depend on η, we could prove the whole family to be PPT directly.

Theorem soundness : $\forall$ (M N:term), M $\approx$ N $\rightarrow$ type0scheme sch $\rightarrow$
indistinguishable ($[\![$M$]\!]$ (shf N M)) ($[\![$N$]\!]$ (shf N M)).

For any two formally equivalent terms M and N, if sch is type-0 then the bitstrings generated by $[\![\cdot]\!]$ on M and N (starting from randomness shf N M) are computationally indistinguishable. In the proof, following [4], we reason by contradiction and assume an adversary A which violates indistinguishability of M and N. We then build two extended adversaries $A_0 \in F(A, sch, E(G_\eta(c_0), \cdot), E_\eta(G_\eta(c_1), \cdot))$ and $A'_0 \in F(A, sch, E(G_\eta(c_0), zero), E_\eta(G_\eta(c_0), zero))$ and prove (1) they are the same up to oracles, and (2) they are PPT, if A is PPT. Then we find a 'large gap' i between $[\![\mathrm{M}_i]\!]$ and $[\![\mathrm{M}_{i-1}]\!]$, (where M_i denotes the term p M' $\{K_1, \ldots, K_i\}$, for M' being (renaming M) and $K_1, \ldots, K_i$ given by ks M i), and prove

$$Pr_\eta[A_0(c_n, zero) = zero] = Pr_\eta[A(c_n, [\![\mathrm{M}_i]\!](c))]$$
$$Pr_\eta[A'_0(c_n, zero) = zero] = Pr_\eta[A(c_n, [\![\mathrm{M}_{i-1}]\!](c))]$$

This crucial step is rigorously proved in our development by using two intermediate lemmas proving that $Pr_\eta[\cdot]$ commutes with (1) substitutions (of renaming) and with (2) patterns p. These lemmas also help into understanding exactly when the property of 'renaming' is used. Finally, our formalized proof was also useful to detect some bad usages of the p function when called with incomplete keys (e.g., as in $M := (\{K_1\}_{K_2}, \{58\}_{K_1})$, where a call $p\ M\ K_2 = (\{K_1\}_{K_2}, \Box)$ should be avoided) if one attempts to deviate from the original proof [4].

5 Conclusions

This development shows the feasibility of formalizing cryptographic soundness results in a general purpose interactive theorem prover like Coq. Moreover, we obtained deep insight into the original, soundness paper-based proof, as well as developed provable cryptographic concepts that can be useful in other cryptographic formalizations, e.g. to build tools for checking game-based cryptographic proofs [8].

Future Work. We already started developing the converse to the soundness theorem, i.e. completeness [11]. In fact this theorem seems easier than the soundness one of this paper, as it involves no reduction, only bitstrings computations and inductive proofs that can be easily dealt with in Coq. Another goal is to model a richer language (in the lines of [3]), or to formalize soundness in an active setting [12]. This would require us to model stateful oracles (i.e. to model protocol participants). This can be done in the current setting e.g. encoding the state as a value "piggybacked" to the attacker, which needs to return it in every oracle invocation.

Coq statistics. Besides the soundness theorem, there are 98 definitions and 65 lemmas in 2853 lines of code. We use Coq 8.0 with ssreflect from G. Gonthier.

Acknowledgements. M. Abadi, C. Fournet, A. Mahboubi, G. Gonthier, E. Tassi, G. Barthe, S. Zanella, R. Janvier, B. Gregoire, J. den Hartog, and P. van Rossum provided useful remarks.

References

[1] Computational soundness of formal encryption in coq (long version). At `http://www.msr-inria.inria.fr/~corin/full.pdf`.

[2] The Coq theorem prover. At `coq.inria.fr`.

[3] M. Abadi and J. Jurjens. Formal eavesdropping and its computational interpretation. In *Fourth International Symposium on Theoretical Aspects of Computer Software (TACS2001)*, LNCS. Springer-Verlag, 2001.

[4] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Journal of Cryptology*, number 15, pages 103–127. Springer-Verlag, 2000.

[5] 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, Cork, Ireland, July 2004. Springer-Verlag.

[6] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 652–663, Lisboa, Portugal, July 2005. Springer.

[7] B. Blanchet. A computationally sound mechanized prover for security protocols. In *IEEE Symposium on Security and Privacy*, pages 140–154, Oakland, California, May 2006.

[8] R. Corin and J. den Hartog. A probabilistic hoare-style logic for game-based cryptographic proofs. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *33rd International Colloquium on Automata, Languages and Programming (ICALP), Venice, Italy, Track C (Security), Part II*, volume 4052 of *LNCS*, pages 252–263. Springer, July 10-14, 2006.

[9] F. D. Garcia and P. van Rossum. Sound computational interpretation of symbolic hashes in the standard model. In H. Yoshiura, K. Sakurai, K. Rannenberg, Y. Murayama, and S. ichi Kawamura, editors, *IWSEC*, volume 4266 of *LNCS*, pages 33–47. Springer, 2006.

[10] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In J. I. Lim and D. H. Lee, editors, *6th Annual Int. Conf. on Information Security and Cryptology (ICISC)*, volume LNCS 2971, pages 55–66, Seoul, Korea, Nov 2003. Springer-Verlag, Berlin.

[11] D. Micciancio and B. Warinschi. Completeness theorems of the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.

[12] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Theory of Cryptography: First Theory of Cryptography Conference (TCC '04)*, volume 2951 of *LNCS*, pages 133–151. Springer, 2004.

[13] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[14] C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner. Cryptographically sound theorem proving. In *CSFW '06: Proceedings of the 19th IEEE Workshop on Computer Security Foundations*, pages 153–166, Washington, DC, USA, 2006. IEEE Computer Society.