Decidability of a Sound Set of Inference Rules for Computational Indistinguishability

Adrien Koutsos

Decidability of a Sound Set of Inference Rules for Computational Indistinguishability

Adrien Koutsos
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay
Cachan, France
adrien.koutsos@lsv.fr

***Abstract*—Computational indistinguishability is a key property in cryptography and verification of security protocols. Current tools for proving it rely on cryptographic game transformations.**

We follow Bana and Comon's approach [1], [2], axiomatizing what an adversary cannot distinguish. We prove the decidability of a set of first-order axioms which are computationally sound, though incomplete, for protocols with a bounded number of sessions whose security is based on an IND-CCA$_2$ encryption scheme. Alternatively, our result can be viewed as the decidability of a family of cryptographic game transformations. Our proof relies on term rewriting and automated deduction techniques.

***Index Terms*—Security Protocols, Automated Deduction, Decision Procedure, Computational Indistinguishability**

I. Introduction

Designing security protocols is notoriously hard. For example, the TLS protocol used to secure most of the Internet connections was successfully attacked several times at the protocol level, e.g. the LogJam attack [3] or the TripleHandshake attack [4]. This shows that, even for high visibility protocols, and years after their design, attacks are still found.

Using formal methods to prove a security property is the best way to get a strong confidence. However, there is a difficulty, which is not present in standard program verification: we need not only to specify formally the program and the security property, but also the attacker. Several attacker models have been considered in the literature.

A popular attacker model, the *Dolev-Yao attacker*, grants the attacker the complete control of the network: he can intercept and re-route all messages. Besides, the adversary is allowed to modify messages using a fixed set of rules (e.g. given a cipher-text and its decryption key, he can retrieve the plain-text message). Formally, messages are terms in a term algebra and the rules are given through a set of rewrite rules. This model is very amenable to automatic verification of security properties. There are several automated tools, such as, e.g., ProVerif [5], Tamarin [6] and Deepsec [7].

Another attacker model, closer to a real world attacker, is the *computational attacker* model. This adversary also controls the network, but this model does not restrict the attacker to a fixed set of operations: the adversary can perform any probabilistic polynomial time computation. More formally, messages are bit-strings, random numbers are sampled uniformly among bit-strings in $\{0,1\}^\eta$ (where η is the *security parameter*) and the attacker is any probabilistic polynomial-time Turing machine (PPTM). This model offers stronger guarantees than the Dolev-Yao model (DY model), but formal proofs are harder to complete and more error-prone. There exist several formal verification tools in this model: for example, EasyCrypt [8] which relies on pRHL, and CryptoVerif [9] which performs game transformations. As expected, such tools are less automatic than the verification tools in the DY model. Moreover, the failure to find a proof in such tools, either because the proof search failed or did not terminate, or because the user could not manually find a proof, does not give any indication on the actual security of the protocol.

There is an alternative approach, the Bana-Comon model (BC model), introduced in [2]. In this model, we express the security of a protocol as the unsatisfiability of a set of formulas in first-order logic. The formulas contain the negation of the security property and *axioms*, which reflect implementation assumptions, such as functional correctness and cryptographic hypotheses on the security primitives. This method has several advantages over pRHL and game transformations. First, it is simpler, as there is no security game and no probabilities, only a first-order formula. Then, carrying out a proof of unsatisfiability in this logic entails the security of the protocol in the computational model. Finally, the absence of such a proof implies the existence of a model of the formula, i.e. an attack, albeit not necessarily a computational one; nonetheless, we know that the security of the protocol *cannot* be obtained without extra assumptions. Note that the Bana-Comon approach is only valid for protocols with a finite number of sessions (there is no unbounded replication). Since this is the model we use, we inherit this restriction.

There is another input to security proofs that we did not discuss yet: the class of security properties considered. Roughly, there are two categories. *Reachability* properties state that some bad state is unreachable. This includes, for example, authentication or (weak) secrecy. *Indistinguishability* properties state that an adversary cannot distinguish between the executions of two protocols. This allows for more complex properties, such as strong secrecy and unlinkability.

a) Deciding Security: When trying to prove a protocol, there are three possible outcomes: either we find a proof, which gives security guarantees corresponding to the attacker model; or we find an attack, meaning that the protocol is insecure; or the tool or the user (for interactive provers) could not carry out the proof and failed to find an attack. The latter case may happen for two different reasons. First, we could

neither find a proof nor an attack because the proof method used is incomplete. In that case, we need either to make new assumptions and try again, or to use another proof technique. Second, the tool may not terminate on the protocol considered. This is problematic, as we do not know if we should continue waiting, and consume more resources and memory, or try another method.

This can be avoided for decidable classes of protocols and properties. Of course, such classes depend on both the attacker model and the security properties considered. We give here a non-exhaustive survey of such results. In the symbolic model, [10] shows decidability of secrecy (a reachability property) for a bounded number of sessions. In [11], the authors show the decidability of a secrecy property for *depth-bounded* protocols, with an unbounded number of sessions, using Well-Structured Transition Systems [12]. Chrétien et al [13] show the decidability of indistinguishability properties for a restricted class of protocols. E.g., they consider processes communicating on distinct channels and without else branches. The authors of [14] show the decidability of symbolic equivalence for a bounded number of sessions, but with conditional branching.

In the computational model, we are aware of only one direct result. In [15], the authors show the decidability of the security of a formula in the BC model, for *reachability properties*, for a bounded number of sessions. But there is an indirect way of getting decidability in the computational model, through a *computational soundness* theorem (e.g. [16]). A computational soundness theorem states that, for some given classes of protocols and properties, symbolic security implies computational security. These results usually make strong implementation assumptions (e.g. parsing assumptions, or the absence of dishonest keys), and require that the security primitives satisfy strong cryptographic hypothesis. By combining a decidability result in the symbolic model with a computational soundness theorem, which applies to the considered classes of protocols and properties (e.g. [17] for reachability properties, or [18] for indistinguishability properties), we obtain a decidability result in the computational model.

We discuss further related works later, in Section VIII.

b) Contributions: In this paper, we consider the BC model for indistinguishability properties [1]. This is a first-order logic in which we design a set of axioms Ax which includes, in particular, axioms for the IND-CCA$_2$ cryptographic assumption [19]. Given a protocol and a security property, we can build, using a folding technique described in [1], a ground atomic formula ψ expressing the security of the protocol. Showing the unsatisfiability of the conjunction of the axioms Ax and the negation of ψ entails the security of the protocol in the computational model, assuming that the encryption scheme is IND-CCA$_2$ secure.

Our main result is the decidability of the problem:

Input: A ground formula $\vec{u} \sim \vec{v}$.

Question: Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?

That is, we show the decidability of a sound, though incomplete, axiomatization of computational indistinguishability.

All the formulas in Ax are Horn clauses, therefore to show the unsatisfiability of $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ we use resolution with a negative strategy: we see axioms in Ax as inference rules and look for a derivation of the goal $\vec{u} \sim \vec{v}$. We prove the decidability of the corresponding satisfiability problem.

The main difficulty lies in dealing with equalities (defined through a term rewriting system R). First we show the completeness of an ordered strategy by commuting rule applications. This allows us to have only one rewriting modulo R at the beginning of the proof. We then bound the size of the terms after this rewriting as follows: we identify a class of proof cuts introducing arbitrary subterms; we give proof cut eliminations to remove them; and finally, we show that cut-free proofs are of bounded size w.r.t. the size of the conclusion.

c) Game Transformations: Our result can be reinterpreted as the decidability of the problem of determining whether there exits a sequence of game transformations [20], [21] that allows to prove the security of a protocol. Indeed, one can associate to every axiom in Ax either a cryptographic assumption or a game transformation.

Each unitary axiom in Ax (an atomic formula) corresponds to an instantiation of the IND-CCA$_2$ game. For instance, in the simpler case of IND-CPA security of an encryption $\{\_\}^{\mathsf{n}}_{\mathsf{pk}}$, no polynomial-time adversary can distinguish between two cipher-texts, even if it chooses the two corresponding plain-texts (here, n is the explicit encryption randomness). Initially, the public key pk is given to the adversary, who computes a pair of plain-texts $g(\mathsf{pk})$: g is interpreted as the adversary's computation. Then the two cipher-texts, corresponding to the encryptions of the first and second components of $g(\mathsf{pk})$, should be indistinguishable. This yields the unitary axiom:

$$\{\pi_1(g(\mathsf{pk}))\}^{\mathsf{n}}_{\mathsf{pk}} \sim \{\pi_2(g(\mathsf{pk}))\}^{\mathsf{n}}_{\mathsf{pk}}$$

Similarly, non-unitary axioms correspond to cryptographic game transformations. E.g., the function application axiom:

$$\vec{u} \sim \vec{v} \rightarrow f(\vec{u}) \sim f(\vec{v})$$

states that if no adversary can distinguish between the arguments of a function call, then no adversary can distinguish between the images. As for a cryptographic game transformation, the soundness of this axiom is shown by reduction. Given a winning adversary $\mathcal{A}$ against the conclusion $f(\vec{u}) \sim f(\vec{v})$, we build a winning adversary $\mathcal{B}$ against $\vec{u} \sim \vec{v}$: the adversary $\mathcal{B}$, on input $\vec{w}$ (which was sampled from $\vec{u}$ or $\vec{v}$), computes $f(\vec{w})$ and then gives the result to the distinguisher $\mathcal{A}$. The advantage of $\mathcal{B}$ against $\vec{u} \sim \vec{v}$ is then the advantage of $\mathcal{A}$ against $f(\vec{u}) \sim f(\vec{v})$, which is (by hypothesis) non negligible.

By interpreting every axiom in Ax as a cryptographic assumption or a game transformation, and the goal formula $\vec{u} \sim \vec{v}$ as the initial game, our result can be reformulated as showing the decidability of the following problem:

Input: An initial game $\vec{u} \sim \vec{v}$.

Question: Is there a sequence of game transformations in Ax showing that $\vec{u} \sim \vec{v}$ is secure?

From this point of view, our result guarantees a kind of sub-formula property for the intermediate games appearing in the game transformation proof. We may only consider intermediate games that are in a finite set computable from the original protocol: the other games are provably unnecessary detours. To our knowledge, our result is the first showing the decidability of a class of game transformations.

d) Scope and Limitations: To achieve decidability, we had to remove or restrict some axioms. The most important restriction is arguably that we do not include the transitivity axiom. The transitivity axiom states that to show that $\vec{u} \sim \vec{v}$, it is sufficient to find a $\vec{w}$ such that $\vec{u} \sim \vec{w}$ and $\vec{w} \sim \vec{v}$. Obviously, this axiom is problematic for decidability, as the vector of term $\vec{w}$ must be guessed, and may be arbitrarily large. Therefore, instead of directly including transitivity, we push it inside the CCA2 axiom schema, by allowing instances of the CCA2 axiom to deal simultaneously with multiple keys and interleaved encryptions. Of course, this is at the cost of a more complex axiom. We do not know if our problem remains decidable when we include the transitivity axiom.

e) Applications: The BC indistinguishability model has been used to analyse RFID protocols [22], a key-wrapping API [23] and an e-voting protocol [24]. Ideally, we would like future case studies to be carried out automatically and machine checked. Because our procedure has a high complexity, it is unclear whether it can be used directly for this. Still, our procedure could be a building block in a tool doing an incomplete but faster heuristic exploration of the proof space.

CRYPTOVERIF and EASYCRYPT are based on game transformations, directly in the former and through the pRHL logic in the latter. Therefore, our result could be used to bring automation to these tools. Of course, both tools allow for more rules. Still, we could identify which game transformations or rules correspond to our axioms, and apply our result to obtain decidability for this subset of game transformations.

f) Outline: We introduce the logic and the axioms in Section II and III. We then state the main result in Section IV, and depict the difficulties of the proof. Finally we sketch the proof: in Section V we show the rule commutations and some cut eliminations; in Section VI we show a normal form for proofs and its properties; and in Section VII we give more cut eliminations and the decision procedure. We discuss in details the related works in Section VIII. For space reasons, most of the proofs are sketched or omitted. The full proofs can be found in the long version of this paper [25].

II. THE LOGIC

We recall here the logic introduced in [1]. In this logic, terms represent messages of the protocol sent over the network, including the adversary's inputs, which are specified using additional function symbols. Formulas are built using the usual Boolean connectives and FO quantifiers, and a single predicate, $\sim$, which stands for indistinguishability. The semantics of the logic is the usual first-order semantics, though we are particularly interested in computational models, in which terms are interpreted as PPTMs, and $\sim$ is interpreted as computational indistinguishability.

This logic is then used as follows: given a protocol and a security property, we can build (automatically) a single formula $\vec{u} \sim \vec{v}$ expressing the security of the protocol. We specify, through a (recursive) set of axioms, what the adversary *cannot* do. This yields a set of axioms Ax. We show that $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ is unsatisfiable, and that the axioms Ax are valid in the computational model. We deduce from this the security of the protocol in the computational model.

A. Syntax

a) Terms: Terms are built upon a set of function symbols $\mathcal{F}$, a countable set of names $\mathcal{N}$ and a countable set of variables $\mathcal{X}$. This is a sorted logic with two sorts $\mathcal{S}_\mathsf{m}, \mathcal{S}_\mathsf{b}$, with $\mathcal{S}_\mathsf{b} \subseteq \mathcal{S}_\mathsf{m}$.

The set $\mathcal{F}$ of function symbols is composed of a countable set of adversarial function symbols $\mathcal{G}$ (representing the adversary computations), and the following function symbols: the pair $\langle \_, \_ \rangle$, projections π_1, π_2, public and private key generation $\mathsf{pk}(\_), \mathsf{sk}(\_)$, encryption with random seed $\{\_\}^{\_}_{\_}$, decryption $\mathsf{dec}(\_,\_)$, $\mathsf{if\_then\_else\_}$, true, false, zero $\mathbf{0}(\_)$ and equality check $\mathsf{eq}(\_,\_)$. We give their types below:

$$\begin{array}{llll} \langle \_, \_ \rangle, \mathsf{dec}(\_,\_) & : \mathcal{S}_\mathsf{m}^2 \to \mathcal{S}_\mathsf{m} & \mathsf{eq}(\_,\_) & : \mathcal{S}_\mathsf{m}^2 \to \mathcal{S}_\mathsf{b} \\ \pi_1, \pi_2, \mathbf{0}, \mathsf{pk}, \mathsf{sk} & : \mathcal{S}_\mathsf{m} \to \mathcal{S}_\mathsf{m} & \{\_\}^{\_}_{\_} & : \mathcal{S}_\mathsf{m}^3 \to \mathcal{S}_\mathsf{m} \\ \mathsf{if\_then\_else\_} & : \mathcal{S}_\mathsf{b} \times \mathcal{S}_\mathsf{m}^2 \to \mathcal{S}_\mathsf{m} & \mathsf{true}, \mathsf{false} & : \to \mathcal{S}_\mathsf{b} \end{array}$$

Moreover all the names in $\mathcal{N}$ have sort $\mathcal{S}_\mathsf{m}$, and each variable in $\mathcal{X}$ comes with a sort. We let $\mathcal{F}_s$ be $\mathcal{F}$ without the $\mathsf{if\_then\_else\_}$ function symbol, and for any subset $\mathcal{S}$ of the union of $\mathcal{F}$, $\mathcal{N}$ and $\mathcal{X}$, we let $\mathcal{T}(\mathcal{S})$ be the set of terms built upon $\mathcal{S}$.

b) Formulas: For every integer n, we have one predicate symbol $\sim_n$ of arity $2n$, which represents equivalence between two vectors of terms of length n. Formulas are then obtained using the usual Boolean connectives and first-order quantifiers.

c) Semantics: We use the classical first-order logic semantics: every sort is interpreted by some domain, and function symbols and predicates are interpreted as, resp., functions of the appropriate domains and relations on these domains.

We focus on a particular class of such models, the *computational models*. We informally describe the properties of a computational model $\mathcal{M}_c$ (a full description is given in [1]):

- $\mathcal{S}_\mathsf{m}$ is interpreted as the set of probabilistic polynomial time Turing machines equipped with a working tape and two random tapes ρ_1, ρ_2 (one for the protocol random values, the other for the adversary random samplings). Moreover its input is of length η (the security parameter). $\mathcal{S}_\mathsf{b}$ is the restriction of $\mathcal{S}_\mathsf{m}$ to machines that return 0 or 1.
- A name $\mathsf{n} \in \mathcal{N}$ is interpreted as a machine that, on input of length η, extracts a word of length η from the first random tape ρ_1. Furthermore we require that different names extract disjoint parts of ρ_1.
- true, false, $\mathbf{0}(\_)$, $\mathsf{eq}(\_,\_)$, and $\mathsf{if\_then\_else\_}$ are interpreted as expected. For instance, $\mathsf{eq}(\_,\_)$ takes two machines M_1, M_2, and returns M such that on input w

and random tapes ρ_1, ρ_2, M returns 1 if $M_1(w, \rho_1, \rho_2) = M_2(w, \rho_1, \rho_2)$ and 0 otherwise. The function symbol $\mathbf{0}$ is interpreted as the function that, on input of length l, returns the bit-string 0^l.

- A function symbol $g \in \mathcal{G}$ with n arguments is interpreted as a function $[\![g]\!]$ such that there is a polynomial-time Turing machine M_g such that for every machines $(m_i)_{i\leq n}$ in the interpretation domains, and for every inputs w, ρ_1, ρ_2:

$$[\![g]\!]\big((m_i)_{i\leq n}\big)(w, \rho_1, \rho_2) = M_g\big((m_i(w, \rho_1, \rho_2))_{i\leq n}, \rho_2\big)$$

  Observe that M_g cannot access directly the tape ρ_1.
- Protocol function symbols are interpreted as deterministic polynomial-time Turing machine. Their interpretations will be restricted using *implementation axioms* later.
- The interpretation of function symbols is lifted to terms: given an assignment σ of the variables of a term t to elements of the appropriate domains, we write $[\![t]\!]^{\sigma}_{\eta,\rho_1,\rho_2}$ the interpretation of the term with respect to η, ρ_1, ρ_2. σ is omitted when empty. We also omit the other parameters when they are irrelevant.
- The predicate $\sim_n$ is interpreted as *computational indistinguishability* $\approx$, defined by $m_1, \ldots, m_n \approx m'_1, \ldots, m'_n$ iff for every PPTM $\mathcal{A}$ with random tape ρ_2:

$$\big|\, \mathbf{Pr}(\rho_1, \rho_2 : \mathcal{A}((m_i(1^\eta, \rho_1, \rho_2))_{1\leq i\leq n}, \rho_2) = 1) \;-\; \mathbf{Pr}(\rho_1, \rho_2 : \mathcal{A}((m'_i(1^\eta, \rho_1, \rho_2))_{1\leq i\leq n}, \rho_2) = 1) \,\big|$$

  is negligible in η (a function is negligible if it is asymptotically smaller than the inverse of any polynomial).
  Moreover, for all ground terms u, v, we write $\mathcal{M}_c \models u \sim v$ when $[\![u]\!] \approx [\![v]\!]$ in $\mathcal{M}_c$.

Example 1. Let $\mathsf{n}_0, \mathsf{n}_1, \mathsf{n} \in \mathcal{N}$ and $g \in \mathcal{F}$ of arity zero. For every computational model $\mathcal{M}_c$:

$$\mathcal{M}_c \models \mathsf{if}\ g()\ \mathsf{then}\ \mathsf{n}_0\ \mathsf{else}\ \mathsf{n}_1 \sim \mathsf{n}$$

Indeed, the term on the left represents the message obtained by letting the adversary choose a branch, and then sampling from n_0 or n_1 accordingly. This is semantically equivalent to directly performing a random sampling, as done on the right.

III. Axioms

We present the axioms Ax, which are of two kinds:

- *structural axioms* represent properties that hold in every computational model. This includes axioms such as the symmetry of $\sim$, or properties of the if_then_else_.
- *implementation axioms* reflect implementation assumptions, such as the functional correctness of the pair and projections (e.g. $\pi_1(\langle u\,,\, v\rangle) = u$), or cryptographic assumptions on the security primitives (e.g. IND-CCA$_2$).

All our axioms Ax are universally quantified Horn clauses. To show the unsatisfiability of $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$, we use resolution with a negative strategy (which is complete, see [26]). As all axioms are Horn clauses, a proof by resolution with a negative strategy can be seen as a proof tree where each node is indexed by the axiom of Ax used at this resolution step. Hence, axioms will be given as inference rules (where variables are implicitly universally quantified).

A. Equality and Structural Axioms

Some notation conventions: we use $\vec{u}$ to denote a vector of terms; and we use an infix notation for $\sim$, writing $\vec{u} \sim \vec{v}$ when $\vec{u}$ and $\vec{v}$ are of the same length.

The equality and structural axioms we present here already appeared in the literature [1], [22], [27], sometimes with slightly different formulations.

a) Equality: Computational indistinguishability is an equivalence relation (i.e. reflexive, symmetric and transitive). But we can observe that it is not a congruence. E.g. take a computational model $\mathcal{M}_c$, we know that two names n and n' are indistinguishable (since they are interpreted as independent uniform random sampling in $\{0,1\}^\eta$), and n is indistinguishable from itself. Therefore:

$$\mathcal{M}_c \models \mathsf{n} \sim \mathsf{n}' \quad \text{and} \quad \mathcal{M}_c \models \mathsf{n} \sim \mathsf{n}$$

But there is a simple PPTM that can distinguish between $\langle \mathsf{n}\,,\, \mathsf{n}\rangle$ and $\langle \mathsf{n}'\,,\, \mathsf{n}\rangle$: simply test whether the two arguments are equal, if so return 1 and otherwise return 0. Then, with overwhelming probability, this machine will guess from which distribution its input was sampled from.

Even though $\sim$ is not a congruence, we can get a congruence from it: if $\mathsf{eq}(s,t) \sim \mathsf{true}$ holds in all models then, using the semantics of $\mathsf{eq}(\_,\_)$, in every computational model $\mathcal{M}_c$, $[\![s]\!]$ and $[\![t]\!]$ are identical except for a negligible number of samplings. Hence we can replace any occurrence of s by t in a formula without changing its semantics with respect to computational indistinguishability.

We use this in our logic as follows: we let $s = t$ be a shorthand for $\mathsf{eq}(s,t) \sim \mathsf{true}$, and we introduce a set of equalities R (given in Fig. 1) and its congruence closure $=_R$. We split R in four sub-parts: R_1 contains the functional correctness assumptions on the pair and encryption; R_2 and R_3 contain, respectively, the homomorphism properties and simplification rules of the if_then_else_; and R_4 allows to change the order in which conditional tests are performed.

We then introduce a recursive set of rules:

$$\frac{\vec{u}, t \sim \vec{v}}{\vec{u}, s \sim \vec{v}}\ R \qquad (s, t \text{ ground terms with } s =_R t)$$

By orienting R_1, R_2, R_3 from left to right, and carefully choosing an orientation for the ground instances of R_4, we obtain a recursive term rewriting system $\to_R$. We have the following theorem (the proof is in the long version [25]):

Theorem 1. *The TRS $\to_R$ is convergent on ground terms.*

b) Structural Axioms: We now give an informal description of the axioms given in Fig. 2. We describe in details the case study axiom CS, which is the most complicated one. It states that in order to show that:

$$\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ b'\ \mathsf{then}\ u'\ \mathsf{else}\ v'$$

it is sufficient to show that the then branches and the else branches are indistinguishable, *when giving to the adversary the value of the conditional* (i.e. b on the left and b' on the right). We can do better, by considering simultaneously several

$$R_1 \left\{ \begin{array}{ll} \pi_i(\langle x_1, x_2\rangle) = x_i & \mathsf{eq}(x,x) = \mathsf{true} \\ \mathsf{dec}(\{x\}^z_{\mathsf{pk}(y)}, \mathsf{sk}(y)) = x & \end{array} \right.$$

$$R_2 \left\{ \begin{array}{l} f(\vec{u}, \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y, \vec{v}) = \\ \quad \mathsf{if}\ b\ \mathsf{then}\ f(\vec{u}, x, \vec{v})\ \mathsf{else}\ f(\vec{u}, y, \vec{v}) \qquad (f \in \mathcal{F}_s) \\ \mathsf{if}\ (\mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c)\ \mathsf{then}\ x\ \mathsf{else}\ y = \\ \quad \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ x\ \mathsf{else}\ y) \end{array} \right.$$

$$R_3 \left\{ \begin{array}{ll} \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ x = x & \\ \mathsf{if}\ \mathsf{true}\ \mathsf{then}\ x\ \mathsf{else}\ y = x & \mathsf{if}\ \mathsf{false}\ \mathsf{then}\ x\ \mathsf{else}\ y = y \\ \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z = \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z & \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z) = \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z & \end{array} \right.$$

$$R_4 \left\{ \begin{array}{l} \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z = \\ \quad \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z) \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ y\ \mathsf{else}\ z) = \\ \quad \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z) \end{array} \right.$$

Fig. 1. $R = R_1 \cup R_2 \cup R_3 \cup R_4$

terms starting with the same conditional. We also allow some terms $\vec{w}$ and $\vec{w}'$ on the left and right to stay untouched:

$$\frac{\vec{w}, b, (u_i)_i \sim \vec{w}', b', (u'_i)_i \qquad \vec{w}, b, (v_i)_i \sim \vec{w}', b', (v'_i)_i}{\vec{w}, (\mathsf{if}\ b\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i)_i \sim \vec{w}', (\mathsf{if}\ b'\ \mathsf{then}\ u'_i\ \mathsf{else}\ v'_i)_i}$$

This is the only axiom with more than one premise. Furthermore we assume that b, b' do not contain conditionals.

We quickly describe the other structural axioms: Perm allows to change the terms order, using the same permutation on both sides of $\sim$; Restr is a strengthening axiom; R allows to replace a term s by any R-equal term t; the function application axiom FA states that to prove that two images are indistinguishable, it is sufficient to show that the arguments are indistinguishable (we restrict this axiom to the case where f is in $\mathcal{F} \setminus \{\mathbf{0}\}$); Sym states that indistinguishability is symmetrical; and Dup states that giving twice the same value to an adversary is equivalent to giving it only once. All the above axioms are computationally valid.

Proposition 1. *The axioms given in Fig. 2 are valid in any computational model in which the functional correctness assumptions R_1 on pairs and encryptions hold.*

Proof. The proof can be found in [1]. ∎

c) Restrictions: As mentioned earlier, we restricted some axioms to achieve decidability. For example, the CS and FA axioms presented above are weaker than the corresponding axioms in [1]: in the CS axiom, we forbid the terms b and b' from containing conditionals; and we do not allow FA applications on the $\mathbf{0}$ function symbols. These are technical restrictions which are used in the proof, but might be unnecessary.

B. Cryptographic Assumptions

We now show how cryptographic assumptions are translated into unitary axioms. In the computational model, the security of a cryptographic primitive is expressed through a game between a challenger and an attacker (which is a PPTM) that tries to break the primitive.

We present here the IND-CCA$_2$ game (for Indistinguishability against Chosen Ciphertexts Attacks, see [19]). First, the challenger computes a public/private key pair $(\mathsf{pk}(\mathsf{n}), \mathsf{sk}(\mathsf{n}))$ (using a nonce n of length η uniformly sampled), and sends $\mathsf{pk}(\mathsf{n})$ to the attacker. The adversary then has access to two oracles: i) a left-right oracle $\mathcal{O}^b_{\mathsf{LR}}(\mathsf{n})$ that takes two messages m_0, m_1 as input and returns $\{m_b\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}$, where b is an internal bit uniformly sampled at the beginning by the challenger and n_r is a fresh nonce; ii) a decryption oracle $\mathcal{O}_{\mathsf{dec}}(\mathsf{n})$ that, given m, returns $\mathsf{dec}(m, \mathsf{sk}(\mathsf{n}))$ if m was not the result of a previous $\mathcal{O}_{\mathsf{LR}}$ oracle query, and length of m zeros otherwise. Remark that the two oracles have a shared memory. For simplicity, we omit the length constraints of these oracles (they can be found the long version [25]) The advantage $\mathrm{Adv}^{\mathsf{CCA2}}_{\mathcal{A}}(\eta)$ of an adversary $\mathcal{A}$ against this game is the probability for $\mathcal{A}$ to guess the bit b:

$$\Big| \mathbf{Pr}\big(\mathsf{n} : \mathcal{A}^{\mathcal{O}^1_{\mathsf{LR}}(\mathsf{n}), \mathcal{O}_{\mathsf{dec}}(\mathsf{n})}(1^\eta) = 1\big) - \mathbf{Pr}\big(\mathsf{n} : \mathcal{A}^{\mathcal{O}^0_{\mathsf{LR}}(\mathsf{n}), \mathcal{O}_{\mathsf{dec}}(\mathsf{n})}(1^\eta) = 1\big) \Big|$$

An encryption scheme is IND-CCA$_2$ if the advantage $\mathrm{Adv}^{\mathsf{CCA2}}_{\mathcal{A}}(\eta)$ of any adversary $\mathcal{A}$ is negligible in η. The IND-CCA$_1$ game is the restriction of this game where the adversary cannot call $\mathcal{O}_{\mathsf{dec}}$ after having called $\mathcal{O}_{\mathsf{LR}}$. An encryption scheme is IND-CCA$_1$ if $\mathrm{Adv}^{\mathsf{CCA1}}_{\mathcal{A}}(\eta)$ is negligible for any adversary $\mathcal{A}$.

a) CCA1 Axiom: Before introducing the CCA2 axioms, we recall informally the CCA1 axioms from [1]. First, we define a syntactic property on secret keys used as a side-condition of the CCA1 axioms:

Definition 1. *For every ground term t, we say that a secret key $\mathsf{sk}(n)$ appears only in* decryption position *in t if it appears only in subterms of t of the form $\mathsf{dec}(\_, \mathsf{sk}(n))$.*

We now define the CCA1 axioms:

Definition 2. CCA1 *is the computable set of unitary axioms:*

$$\vec{w}, t[\{u\}^{\mathsf{n}_r}_{\mathsf{pk}(n)}] \sim \vec{w}, t[\{v\}^{\mathsf{n}_r}_{\mathsf{pk}(n)}]$$

where: n_r does not appear in $t, u, v, \vec{w}$; n appears only in $\mathsf{pk}(n)$ or $\mathsf{sk}(n)$ in $t, u, v, \vec{w}$; $\mathsf{sk}(n)$ does not appear in $t, \vec{w}$; $\mathsf{sk}(n)$ appears only in decryption position in u, v; and the terms u and v are always of the same length.

Proposition 2. CCA1 *is valid in every computational model where the encryption scheme interpretation is* IND-CCA$_1$.

Proof. (*sketch*) The proof is a reduction that, given a PPTM $\mathcal{A}$ that can distinguish between $\vec{w}, t[\{u\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]$ and $\vec{w}, t[\{v\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]$, builds a winning adversary against the IND-CCA$_1$ game.

We define the adversary. First, it computes $[\![u]\!]$ and $[\![v]\!]$, calling the decryption oracle if necessary. It then sends them to the challenger who answers c, which is either $[\![\{u\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]\!]$ or $[\![\{v\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]\!]$. Observe that we need the freshness hypothesis on n_r as it is drawn by the challenger and the adversary cannot sample it. Using c, the adversary computes $[\![t[c]]\!]$, which it can do because the secret key does not appear in t, and then returns the bit $\mathcal{A}([\![t[c]]\!])$. The advantage of the adversary is exactly the advantage of $\mathcal{A}$, which we assumed non-negligible, hence the adversary wins the game. ∎

$$\frac{u_{\pi(1)},\ldots,u_{\pi(n)} \sim v_{\pi(1)},\ldots,v_{\pi(n)}}{u_1,\ldots,u_n \sim v_1,\ldots,v_n}\ \mathsf{Perm} \qquad \frac{\vec{u},t \sim \vec{v},t'}{\vec{u} \sim \vec{v}}\ \mathsf{Restr} \qquad \text{for any } s =_R t,\ \frac{\vec{u},t \sim \vec{v}}{\vec{u},s \sim \vec{v}}\ R \qquad \frac{\vec{u}_1,\vec{v}_1 \sim \vec{u}_2,\vec{v}_2}{f(\vec{u}_1),\vec{v}_1 \sim f(\vec{u}_2),\vec{v}_2}\ \mathsf{FA}$$

$$\frac{\vec{u},t \sim \vec{v},t'}{\vec{u},t,t \sim \vec{v},t',t'}\ \mathsf{Dup} \qquad \frac{\vec{v} \sim \vec{u}}{\vec{u} \sim \vec{v}}\ \mathsf{Sym} \qquad \text{for any } b,b' \in \mathcal{T}(\mathcal{F}_s,\mathcal{N}),\ \frac{\vec{w},b,(u_i)_i \sim \vec{w}',b',(u'_i)_i \qquad \vec{w},b,(v_i)_i \sim \vec{w}',b',(v'_i)_i}{\vec{w},(\mathsf{if}\ b\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i)_i \sim \vec{w}',(\mathsf{if}\ b'\ \mathsf{then}\ u'_i\ \mathsf{else}\ v'_i)_i}\ \mathsf{CS}$$

Conventions: π is a permutation of $\{1,\ldots,n\}$ and $f \in \mathcal{F}\backslash\{\mathbf{0}\}$.

Fig. 2. The Axioms Struct-Ax.

Remark 1. In the CCA1 axiom, we did not specify how we ensure that u and v are always of the same length. Since the length of a term depends on implementation details (e.g. how the pair $\langle \_, \_ \rangle$ implemented), we let the user supply implementation assumptions, but require that these assumptions satisfy some properties (this is necessary to get decidability). To simplify the presentation, we omit all length constraints in the rest of this paper. We explain how they are handled in the long version [25].

b) CCA2 *Axiom:* To extend this axiom to the IND-CCA$_2$ game, we need to deal with calls to the decryption oracle performed after some calls to the left-right oracle. For example, consider the case where one call (u,v) was made. Let $\alpha \equiv \{u\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$ and $\alpha' \equiv \{v\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$ (where $\equiv$ denotes syntactic equality) be the result of the call on, respectively, the left and the right. A naive first try could be to state that decryptions are indistinguishable. That is, if we let $s \equiv t[\alpha]$ and $s' \equiv t[\alpha']$, then $\mathsf{dec}(s,\mathsf{sk(n)}) \sim \mathsf{dec}(s',\mathsf{sk(n)})$. But this is not valid: for example, take $u \equiv 0, v \equiv 1$, $t \equiv g([])$ (where [] is a hole variable). Then the adversary can, by interpreting g as the identity function, obtain a term semantically equal to 0 on the left and 1 on the right. This allows him to distinguish between the left and right cases.

We prevent this by adding a guard checking that we are not decrypting α on the left (resp. α' on the right): if not, we return the decryption $\mathsf{dec}(s,\mathsf{sk(n)})$ (resp. $\mathsf{dec}(s',\mathsf{sk(n)})$) asked for, otherwise we return a dummy message $\mathbf{0}(\mathsf{dec}(s,\mathsf{sk(n)}))$ (resp. $\mathbf{0}(\mathsf{dec}(s',\mathsf{sk(n)}))$).

Definition 3. CCA2^s *is the (recursive) set of unitary axioms:*

$$\begin{aligned} \vec{w},t[\alpha],\ &\mathit{if}\ \mathsf{eq}(s,\alpha)\ \mathit{then}\ \mathbf{0}(\mathsf{dec}(t[\alpha],\mathsf{sk}(n)))\\ &\qquad \mathit{else}\ \mathsf{dec}(t[\alpha],\mathsf{sk}(n))\\ \sim \vec{w},t[\alpha'],&\mathit{if}\ \mathsf{eq}(s',\alpha')\ \mathit{then}\ \mathbf{0}(\mathsf{dec}(t[\alpha'],\mathsf{sk}(n)))\\ &\qquad \mathit{else}\ \mathsf{dec}(t[\alpha'],\mathsf{sk}(n)) \end{aligned}$$

under the side-conditions of Definition 2.

This axiom is valid whenever the encryption is IND-CCA$_2$.

Proposition 3. CCA2^s *is valid in every computational model where the encryption scheme interpretation is* IND-CCA$_2$.

This construction can be generalized to any number of calls to the left-right oracle, by adding a guard for each call, and to any number of keys. The general CCA2 axioms can be found in the long version [25]. [1] Still, a few comments: we use extra syntactic side-conditions to remove superfluous guards; we allow for α-renaming of names; we restrict t to be without if_then_else_ and $\mathbf{0}$; the axioms allow for an arbitrary number of public/private key pairs to be used simultaneously; and finally, an instance of the axiom can contain any number of interleaved left-right and decryption oracles calls.

Remark 2. The last point is what allows us to avoid transitivity in proofs. For example, consider four encryptions, two of them (α and γ) using the public key $\mathsf{pk(n)}$, and the other two (β and δ) using the public key $\mathsf{pk(n')}$:

$$\alpha \equiv \{A\}^{\mathsf{n}_0}_{\mathsf{pk(n)}} \quad \beta \equiv \{B\}^{\mathsf{n}_1}_{\mathsf{pk(n')}} \quad \gamma \equiv \{C\}^{\mathsf{n}_0}_{\mathsf{pk(n)}} \quad \delta \equiv \{D\}^{\mathsf{n}_1}_{\mathsf{pk(n')}}$$

Then the following formula is a valid instance of the CCA2 axioms on, simultaneously, keys $\mathsf{pk(n)}$ and $\mathsf{pk(n')}$:

$$\frac{}{\alpha,\beta \sim \gamma,\delta}\ \mathsf{CCA2}(\mathsf{pk(n)},\mathsf{pk(n')})$$

However, proving the above formula using CCA2 only on one key at a time, as in [1], requires transitivity:

$$\frac{\dfrac{}{\alpha,\beta \sim \alpha,\delta}\ \mathsf{CCA2}(\mathsf{pk(n')}) \qquad \dfrac{}{\alpha,\delta \sim \gamma,\delta}\ \mathsf{CCA2}(\mathsf{pk(n)})}{\alpha,\beta \sim \gamma,\delta}$$

C. Comments and Examples

Our set of axioms is not complete w.r.t. the computational interpretation semantics. Indeed, being so would mean axiomatizing exactly which distributions (computable in polynomial time) can be distinguished by PPTMs, which is unrealistic and would lead to undecidability. E.g., if we completely axiomatized IND-CCA$_2$, then showing the satisfiability of our set of axioms would show the existence of IND-CCA$_2$ functions, which is an open problem.

Still, our axioms are expressive enough to complete concrete proofs of security. We illustrate this with two simple examples: a proof of the formula in Example 1, and a proof of the security of one round of the NSL protocol [28]. Of course, such proofs can be found automatically using our decision procedure.

Example 2. We give a proof of the formula of Example 1:

$$\mathsf{if}\ g()\ \mathsf{then}\ \mathsf{n}_0\ \mathsf{else}\ \mathsf{n}_1 \sim \mathsf{n}$$

First, we introduce a conditional $g()$ on the right to match the structure of the left side using R. Then, we split the proof

[1] Note that axioms for the IND-CCA$_2$ cryptographic assumption have already appeared in the literature, in [27]. These axioms are only for a single call to the left-right oracle, and a single key. Our axiom schema is more general.

in two using the CS axiom. We conclude using the reflexivity modulo α-renaming axiom (this axiom is subsumed by CCA2, therefore we do not include it in AX).

$$\dfrac{\dfrac{\dfrac{}{g(),\mathsf{n}_0 \sim g(),\mathsf{n}}\ \text{REFL} \qquad \dfrac{}{g(),\mathsf{n}_1 \sim g(),\mathsf{n}}\ \text{REFL}}{\text{if } g() \text{ then } \mathsf{n}_0 \text{ else } \mathsf{n}_1 \sim \text{if } g() \text{ then n else n}}\ \text{CS}}{\text{if } g() \text{ then } \mathsf{n}_0 \text{ else } \mathsf{n}_1 \sim \mathsf{n}}\ R$$

a) Proof of NSL: We consider a simple setting with one initiator A, one responder B and no key server. An execution of the NSL protocol is given in Fig. 3.

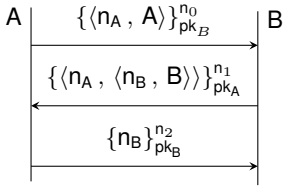


Fig. 3. The NSL protocol.

We write this in the logic. First, we let $\mathsf{pk_A} \equiv \mathsf{pk}(\mathsf{n_A})$ and $\mathsf{sk_A} \equiv \mathsf{sk}(\mathsf{n_A})$ be the public/private key pair of agent A (we define similarly ($\mathsf{pk_B}$, $\mathsf{sk_B}$)). Since A does not wait for any input before sending its first message, we put it into the initial frame:

$$\phi_0 \equiv \mathsf{pk_A}, \mathsf{pk_B}, \{\langle \mathsf{n_A}, \mathsf{A}\rangle\}^{\mathsf{n_0}}_{\mathsf{pk}_B}$$

Then, both agents wait for a message before sending a single reply. When receiving $\mathbf{x_A}$ (resp. $\mathbf{x_B}$), the answer of agent A (resp. B) is expressed in the logic as follows:

$$\begin{aligned} t_{\mathsf{A}}[\mathbf{x_A}] \equiv\ & \text{if } \mathsf{eq}(\pi_1(\mathsf{dec}(\mathbf{x_A}, \mathsf{sk_A})), \mathsf{n_A}) \quad \text{then} \\ & \text{if } \mathsf{eq}(\pi_2(\pi_2(\mathsf{dec}(\mathbf{x_A}, \mathsf{sk_A}))), \mathsf{B}) \text{ then} \\ & \{\pi_1(\pi_2(\mathsf{dec}(\mathbf{x_A}, \mathsf{sk_A})))\}^{\mathsf{n_2}}_{\mathsf{pk_B}} \\ t_{\mathsf{B}}[\mathbf{x_B}] \equiv\ & \text{if } \mathsf{eq}(\pi_2(\mathsf{dec}(\mathbf{x_B}, \mathsf{sk_B})), \mathsf{A}) \quad \text{then} \\ & \{\langle \pi_1(\mathsf{dec}(\mathbf{x_B}, \mathsf{sk_B})), \langle \mathsf{n_B}, \mathsf{B}\rangle\rangle\}^{\mathsf{n_1}}_{\mathsf{pk_A}} \end{aligned}$$

During an execution of the protocol, the adversary has several choices. First, it decides whether to interact first with A or B. We focus on the case where it first sends a message to B (the other case is similar). Then, it can honestly forward the messages or forge new ones. E.g., when sending the first message to B, it can either forward A's message $\{\langle \mathsf{n_A}, \mathsf{A}\rangle\}^{\mathsf{n_0}}_{\mathsf{pk}_B}$ or forge a new message. We are going to prove the security of the protocol in the following case (the other cases are similar):

- the first message, sent to B, is honest. Therefore we take $\mathbf{x_B} \equiv \{\langle \mathsf{n_A}, \mathsf{A}\rangle\}^{\mathsf{n_0}}_{\mathsf{pk}_B}$, and the answer from B is:

$$t_{\mathsf{B}}[\mathbf{x_B}] =_R \{\langle \mathsf{n_A}, \langle \mathsf{n_B}, \mathsf{B}\rangle\rangle\}^{\mathsf{n_1}}_{\mathsf{pk_A}}$$

- the second message, sent to A, is forged. Therefore we take $\mathbf{x_A} \equiv g(\phi_1)$, where $\phi_1 \equiv \phi_0, t_{\mathsf{B}}[\mathbf{x_B}]$. As, a priori, nothing prevents $g(\phi_1)$ from being equal to $t_{\mathsf{B}}[\mathbf{x_B}]$, we use the conditional $\mathsf{eq}(g(\phi_1), t_{\mathsf{B}}[\mathbf{x_B}])$ to ensure that this message is forged. The answer from A is then:

$$s \equiv \text{if } \mathsf{eq}(g(\phi_1), t_{\mathsf{B}}[\mathbf{x_B}]) \text{ then } 0 \text{ else } t_{\mathsf{A}}[g(\phi_1)] \qquad (1)$$

We show the secrecy of the nonce $\mathsf{n_B}$: we let $t'_{\mathsf{B}}[\mathbf{x_B}]$ (resp. s') be the term $t_{\mathsf{B}}[\mathbf{x_B}]$ (resp. s) where we replaced all occurrences of $\mathsf{n_B}$ by 0. For example, $t'_{\mathsf{B}}[\mathbf{x_B}] =_R \{\langle \mathsf{n_A}, \langle 0, \mathsf{B}\rangle\rangle\}^{\mathsf{n_1}}_{\mathsf{pk_A}}$. This yields the following goal formula:

$$\phi_0, t_{\mathsf{B}}[\mathbf{x_B}], s \sim \phi_0, t'_{\mathsf{B}}[\mathbf{x_B}], s' \qquad (2)$$

Remark 3. The process of computing the formula from the protocol description can be done automatically, using a simple procedure similar to the folding procedure from [1]. The formula in (2) has already been split between the honest and dishonest cases using the case study axiom CS (we omit the CS applications to keep the proof readable). For example, the term in (1) is the "else" branch of a CS application on conditional $\mathsf{eq}(g(\phi_1), t_{\mathsf{B}}[\mathbf{x_B}])$ (which does not contain nested conditionals, as required by the CS side-condition).

We now proceed with the proof. We let δ be the guarded decryption that will be used in the CCA2 axiom:

$$\begin{aligned} \delta \equiv\ & \text{if } \mathsf{eq}(g(\phi_1), t_{\mathsf{B}}[\mathbf{x_B}]) \text{ then } \mathbf{0}(\mathsf{dec}(g(\phi_1), \mathsf{sk_A})) \\ & \text{else } \mathsf{dec}(g(\phi_1), \mathsf{sk_A}) \end{aligned} \qquad (3)$$

and s_δ be the term s where all occurrences of $\mathsf{dec}(g(\phi_1), \mathsf{sk_A})$ have been replaced by δ. We have $s =_R s_\delta$. We also introduce shorthands for some subterms of s_δ: we let a_δ, b_δ and e_δ be the terms $\mathsf{eq}(\pi_1(\delta), \mathsf{n_A})$, $\mathsf{eq}(\pi_2(\pi_2(\delta))), \mathsf{B})$ and $\{\pi_1(\pi_2(\delta))\}^{\mathsf{n_2}}_{\mathsf{pk_B}}$. We define $\delta', s'_{\delta'}, a'_{\delta'}, b'_{\delta'}$ and $e'_{\delta'}$ similarly.

We then rewrite s and s' into s_δ and $s'_{\delta'}$ using R. Then we apply FA several times, first to deconstruct s_δ and $s'_{\delta'}$, and then to deconstruct a_δ, b_δ and $a'_{\delta'}, b'_{\delta'}$. Finally, we use Dup to remove duplicates, and we apply CCA2 simultaneously on key pairs $(\mathsf{pk_A}, \mathsf{sk_A})$ and $(\mathsf{pk_B}, \mathsf{sk_B})$ (we omit here the details of the syntactic side-conditions that have to be checked):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\phi_0, t_{\mathsf{B}}[\mathbf{x_B}], \mathsf{n_A}, \delta, e_\delta \sim \phi_0, t'_{\mathsf{B}}[\mathbf{x_B}], \mathsf{n_A}, \delta', e'_{\delta'}}\ \text{CCA2}}{\phi_0, t_{\mathsf{B}}[\mathbf{x_B}], a_\delta, b_\delta, e_\delta \sim \phi_0, t'_{\mathsf{B}}[\mathbf{x_B}], a'_{\delta'}, b'_{\delta'}, e'_{\delta'}}\ (\text{FA}, \text{Dup})^*}{\phi_0, t_{\mathsf{B}}[\mathbf{x_B}], s_\delta \sim \phi_0, t'_{\mathsf{B}}[\mathbf{x_B}], s'_{\delta'}}\ (\text{FA}, \text{Dup})^*}{\phi_0, t_{\mathsf{B}}[\mathbf{x_B}], s \sim \phi_0, t'_{\mathsf{B}}[\mathbf{x_B}], s'}\ R$$

IV. Main Result and Difficulties

We let Ax be the conjunction of Struct-Ax and CCA2. We now state the main result of this paper.

Theorem (Main Result)**.** *The following problem is decidable:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?*

We give here an overview of the problems that have to be overcome in order to obtain the decidability result. Before starting, a few comments. We close all rules under permutations. The Sym rule commutes with all the other rules, and the CCA2 unitary axioms are closed under Sym. Therefore we can remove Perm and Sym from the set of rules. Observe that CS, FA, Dup and CCA2 are all *decreasing rules*, i.e. the premises are smaller than the conclusion. The only non-decreasing rules are R, which may rewrite a term into a larger one, and Restr, which we eliminate later. Therefore we now focus on R.

a) Necessary Introductions: As we saw in Example 2, it might be necessary to use R in the "wrong direction", typically to introduce new conditionals. A priori, this yields an unbounded search space. Therefore our goal is to characterize in which situations we need to use R in the "wrong direction", and with which instances. We identify two necessary reasons for introducing new conditionals.

First, to match the shape of the term on the other side, like $g()$ in Example 2. In this case, the introduced conditional is exactly the conditional that appeared on the other side of $\sim$. With more complex examples this may not be the case. Nonetheless, an introduced conditional is always bounded by the conditional it matches.

Second, we might introduce a guard in order to fit to the definition of safe decryptions in the CCA2 axioms, as in (3). Here also, the introduced guard will be of bounded size. Indeed, guards of $\mathsf{dec}(s, \mathsf{sk})$ are of the form $\mathsf{eq}(s, \alpha)$ where α is a subterm of s. Therefore, for a fixed s, there are a bounded number of them, and they are of bounded size.

Example 3 (Cut Elimination). These conditions are actually sufficient. We illustrate this on an example where the CS rule is applied on two conditionals that have just been introduced.

$$\dfrac{\dfrac{a, s \sim b, t \qquad a, s \sim b, t}{\mathsf{if}\ a\ \mathsf{then}\ s\ \mathsf{else}\ s \sim \mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ t}\ \mathsf{CS}}{s \sim t}\ R$$

Here a and b can be of arbitrary size. Intuitively, this is not a problem since any proof of $a, s \sim b, t$ includes a proof of $s \sim t$. Formally, we have the following weakening lemma.

Lemma 1. *For every proof P of a ground formula $\vec{u}, s \sim \vec{v}, t$, there exists a proof P' of $\vec{u} \sim \vec{v}$ where P' is no larger than P.*

Proof. (*sketch*) The full proof is in the long version [25]. We prove by induction on P that the Restr rule is admissible using $\mathsf{Ax}\backslash\{\mathsf{Restr}\}$. For this to work, we need the CCA2 axioms to be closed under Restr. Note that this creates some problems, which are dealt with in [25]. ∎

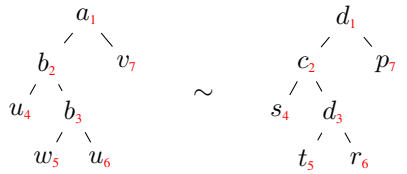
Using this lemma, we can deal with Example 3 by doing a proof cut elimination. More generally, by induction on the proof size, we can guarantee that no such proof cuts appear.

This is the strategy we are going to follow: look for proof cuts that introduce unbounded new terms, eliminate them, and show that after sufficiently many cut eliminations all the subterms appearing in the proof are bounded by the (R-normal form of the) conclusion.

But a proof may contain more complex behaviors than just the introduction of a conditional followed by a CS application. For example the conditional being matched could have been itself introduced earlier to match another conditional, which itself was introduced to match a third conditional etc.

Example 4. We illustrate this on an example. When it is more convenient, we write terms containing only if_then_else_ and other subterms (handled as constants) as binary trees; we also index some subterms with a number, which helps keeping track of them across rule applications.

$$\dfrac{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}{\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t}\ R$$



where $p_7 \equiv \mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t$. Here the conditionals b, d and the terms w, r are, a priori, arbitrary. Therefore we would like to bound them or remove them through a cut elimination. The cut elimination technique used in Example 3 does not apply here because we cannot extract a proof of $a \sim c$.

But we can extract a proof of $b_2, b_3 \sim c_2, d_3$. Using Proposition 1, this means that in every appropriate computational model, $[\![b, b]\!] \approx [\![c, d]\!]$. It means that no adversary can distinguish between getting twice the same value sampled from $[\![b]\!]$ and getting a pair of values sampled from $[\![c, d]\!]$. In particular, this means that $[\![c]\!]_{\eta,\rho} = [\![d]\!]_{\eta,\rho}$, except for a negligible number of random tapes ρ.

b) A First Key Lemma: A natural question is to ask whether this semantic equality $[\![c]\!] = [\![d]\!]$ implies a syntactic equality. While this is not the case in general, there are fragments of our logic in which this holds. We annotate the rules FA by the function symbol involved, and we let $\mathsf{FA_s} = \{\mathsf{FA}_f \mid f \in \mathcal{F}_s\}$.

Definition 4. *Let Σ be the set of axiom names, seen as an alphabet. For all $\mathcal{L} \subseteq \Sigma^*$, we let $\mathfrak{F}(\mathcal{L})$ be the fragment of our logic defined by: a formula ϕ is in the fragment iff there exists a proof P such that $P \vdash \phi$ and, for every branch ρ of P, the word w obtained by collecting the axiom names along ρ (starting from the root) is in $\mathcal{L}$.*

Lemma 2. *For all b, b', b'', if $b, b \sim b', b''$ is in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$ then $b' \equiv b''$.*

Proof. The proof relies on the shape of the CCA2 axioms, and can be found in the long version [25]. ∎

Using this lemma, we can deal with Example 4 if $a_1, b_2, b_3 \sim d_1, c_2, d_3$ lies in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$. Using a first time the lemma on $b_2, b_3 \sim c_2, d_3$ we obtain $c \equiv d$, and using again the lemma on $a_1, b_2 \sim d_1, c_2$ (since $d \equiv c$) we deduce $a \equiv b$. Hence the cut elimination introduced before applies.

c) Proof Sketch: We now state the sketch of the proof:

- **Commutations:** first we show that we can assume that rules are applied in some given order. We prove this by showing some commutation results and adding new rules.
- **Proof Cut Eliminations:** through proof cut eliminations, we guarantee that every conditional appearing in the proof is α-bounded. Intuitively a conditional is α-bounded if it is a subterm of the conclusion or if it guards a decryption appearing in an α-bounded term.

- **Decision Procedure:** we give a procedure that, given a goal formula $t \sim t'$, computes the set of α-bounded terms for this formula. We show that this procedure computes a finite set, and deduce that the proof search is finite. This yields an effective algorithm to decide our problem.

V. Commutations and Cut Eliminations

In this section we show, through rule commutations, that we can restrict ourselves to proofs using rules in some given order. Then, we show how this restricts the shapes of the terms.

A. Rule Commutations

Everything in this subsection applies to any set U of unitary axioms closed under Restr. We specialize to CCA2 later.

We start by showing a set of rule commutations of the form $w \Rightarrow w'$, where w and w' are words over the set of rule names. An entry $w \Rightarrow w'$ means that a derivation in w can be rewritten into a derivation in w', with the same conclusion and premises. Here are the basic commutations we use:

$\mathsf{Dup} \cdot R$	$\Rightarrow$	$R \cdot \mathsf{Dup}$
$\mathsf{Dup} \cdot \mathsf{FA}$	$\Rightarrow$	$\mathsf{FA}^* \cdot \mathsf{Dup}$
$\mathsf{Dup} \cdot \mathsf{CS}$	$\Rightarrow$	$\mathsf{CS} \cdot \mathsf{Dup}$

$\mathsf{FA} \cdot R$	$\Rightarrow$	$R \cdot \mathsf{FA}$
$\mathsf{FA} \cdot \mathsf{CS}$	$\Rightarrow$	$R \cdot \mathsf{CS} \cdot \mathsf{FA}$

Lemma 3. *All the above rule commutations are correct.*

Proof. We show only $\mathsf{FA} \cdot R \Rightarrow R \cdot \mathsf{FA}$ (the full proof is in the long version [25]):

$$\dfrac{\dfrac{\vec{u}_1, \vec{v}_1 \sim \vec{u}'_1, \vec{v}'_1}{\vec{u}, \vec{v} \sim \vec{u}', \vec{v}'}\, R}{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}\, \mathsf{FA} \quad \Rightarrow \quad \dfrac{\dfrac{\vec{u}_1, \vec{v}_1 \sim \vec{u}'_1, \vec{v}'_1}{\vec{u}_1, f(\vec{v}_1) \sim \vec{u}'_1, f(\vec{v}'_1)}\, \mathsf{FA}}{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}\, R$$

■

Using these rules, we obtain a first restriction.

Lemma 4. *The ordered strategy* $\mathfrak{F}((\mathsf{CS} + R)^* \cdot \mathsf{FA}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$ *is complete for* $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{U})^*)$.

Proof. First, we commute all the Dup to the right, which yields $\mathfrak{F}((\mathsf{CS} + R + \mathsf{FA})^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$. Then, we commute all FA to the right, stopping at the first Dup. ■

a) Splitting the FA *Rule:* To go further, we split FA as follows: if the deconstructed symbol is if_then_else_ then we denote the function application by $\mathsf{FA}(b, b')$, where b, b' are the involved conditionals; if the deconstructed symbol f is in $\mathcal{F}_s$, then we denote the function application by FA_f. We give below the two new rules:

$$\dfrac{\vec{w}, a, u, v \sim \vec{r}, b, s, t}{\vec{w}, \mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \vec{r}, \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t}\, \mathsf{FA}(b, b') \qquad \dfrac{\vec{u}, \vec{v} \sim \vec{s}, \vec{t}}{\vec{u}, f(\vec{v}) \sim \vec{s}, f(\vec{t})}\, \mathsf{FA}_f$$

The set of rule names is now infinite, since there exists one rule $\mathsf{FA}(b, b')$ for every pair of ground terms b, b'.

b) Further Commutations: Intuitively, we want to use R at the beginning of the proof only. This is helpful since, as we observed earlier, all the other rules are decreasing (i.e. premises are smaller than the conclusion). The problem is that we cannot fully commute CS and R. For example, in:

$$\dfrac{\dfrac{a', u' \sim b', s'}{a, u \sim b, s}\, R \qquad \dfrac{a'', v' \sim b'', t'}{a, v \sim b, t}\, R}{\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t}\, \mathsf{CS}$$

we can commute the rewritings on u, v, s and t, but not on a and b because they appear twice in the premises, and a' and a'' may be different (same for b' and b'').

c) New Rules: We handle this problem by adding new rules to track relations between branches. We give only simplified versions here, the full rules are in the long version [25]. For every a, c in $\mathcal{T}(\mathcal{F}_s, \mathcal{N})$ in R-normal form, we have the rules:

$$\dfrac{\vec{u}, C\left[\boxed{a}\boxed{a}_a\right] \sim \vec{v}, C'\left[\boxed{c}\boxed{c}_c\right]}{\vec{u}, C[a] \sim \vec{v}, C'[c]}\, \mathsf{2Box^s}$$

$$\dfrac{a_1, u \sim c_1, s \qquad a_2, v \sim c_2, t}{\mathsf{if}\ \boxed{a_1}\boxed{a_2}_a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ \boxed{c_1}\boxed{c_2}_c\ \mathsf{then}\ s\ \mathsf{else}\ t}\, \mathsf{CS}^{\mathsf{s}}_{\square}$$

where $\boxed{\ }\boxed{\ }_a$ is a new symbol of sort $\mathcal{S}_{\mathsf{b}}^2 \to \mathcal{S}_{\mathsf{b}}$, and of fixed semantics: it ignores its arguments and has the semantics $[\![a]\!]$. Intuitively, $\boxed{a_1}\boxed{a_2}_a$ stands for the conditional a, and a_1, a_2 are, respectively, the left and right versions of a.

Remark that for the $\mathsf{CS}_{\square}$ rule to be sound we need $[\![a_1]\!]$, $[\![a_2]\!]$ and $[\![a]\!]$ to be equal, up to a negligible number of samplings (same for c_1, c_2 and c). This is not enforced by the rules, so it has to be an invariant of our strategy. We denote $\mathcal{B}$ the set of new function symbols. We need the functions in $\mathcal{B}$ to block the if-homomorphism to ensure that for all $\boxed{a}\boxed{c}_b \in \mathsf{st}(t)$, $[\![a]\!] = [\![c]\!] = [\![b]\!]$. Therefore the TRS R_2 is *not* extended to $\mathcal{B}$. For example we have:

$$\boxed{\mathsf{if}\ a\ \mathsf{then}\ c\ \mathsf{else}\ d}\boxed{e}_b \not\to_R^* \mathsf{if}\ a\ \mathsf{then}\ \boxed{c}\boxed{e}_b\ \mathsf{else}\ \boxed{d}\boxed{e}_b$$

The R rule is replaced by $R_{\square}$ which has an extra side-condition. $R_{\square}$ can rewrite $u[s]$ into $u[t]$ as long as:

$$\left\{\boxed{a}\boxed{c}_b \in \mathsf{st}(t)\right\} \subseteq \left\{\boxed{a}\boxed{c}_b \in \mathsf{st}(u[s])\right\}$$

This ensures that no new arbitrary $\boxed{a}\boxed{c}_b$ is introduced. New boxed conditionals are only introduced through the 2Box rule. Similarly, the FA axiom is *not* extended to $\mathcal{B}$.

Definition 5. *A term t is* well-formed *if for every $\boxed{a}\boxed{c}_b \in \mathsf{st}(t)$, $a =_R c =_R b$. We lift this to formulas as expected.*

Proposition 4. *The following rules preserve well-formedness:*

$$R_{\square}, \mathsf{2Box}, \mathsf{CS}_{\square}, \mathsf{FA_s}, \{\mathsf{FA}(b, b')\}, \mathsf{Dup}$$

Besides, $R_{\square}$, $\mathsf{CS}_{\square}$ and 2Box *are sound on well-formed formulas.*

Proof. The only rule not obviously preserving well-formedness is $R_{\square}$, but its side-conditions guarantee the well-formedness invariant. The only rule that is not always sound is $\mathsf{CS}_{\square}$, and it is trivially sound on well-formed formulas. ■

d) Ordered Strategy: We have new rule commutations.

$$\begin{array}{|lcl|}\hline \mathsf{FA_s}\cdot\mathsf{FA}(b,b') & \Rightarrow & R\cdot\mathsf{FA}(b,b')\cdot{\mathsf{FA_s}}^*\cdot\mathsf{Dup}\\ \hline \mathsf{CS}_\Box\cdot R_\Box & \Rightarrow & R_\Box\cdot\mathsf{CS}_\Box\\ \mathsf{CS}_\Box\cdot\mathsf{2Box} & \Rightarrow & R_\Box\cdot\mathsf{2Box}\cdot\mathsf{CS}_\Box\\ \hline\end{array}$$

Lemma 5. *All the rule commutations above are correct.*

Proof. The proof can be found in the long version [25]. ∎

This allows to have $R_\Box$ rules only at the beginning of the proof.

Lemma 6. *The ordered strategy:*

$$\mathfrak{F}((\mathsf{2Box}+R_\Box)^*\cdot\mathsf{CS}_\Box^*\cdot\{\mathsf{FA}(b,b')\}^*\cdot{\mathsf{FA_s}}^*\cdot{\mathsf{Dup}}^*\cdot\mathsf{U})$$

is complete for $\mathfrak{F}((\mathsf{CS}+\mathsf{FA}+R+\mathsf{Dup}+\mathsf{U})^*)$.

Proof. We start from the result of Lemma 4, split the FA rules and commute rules until we get:

$$\mathfrak{F}((\mathsf{CS}+R)^*\cdot\{\mathsf{FA}(b,b')\}^*\cdot{\mathsf{FA_s}}^*\cdot{\mathsf{Dup}}^*\cdot\mathsf{U})$$

We then replace all applications of CS by $\mathsf{2Box}.\mathsf{CS}_\Box$. All $\boxed{a}\boxed{a}_a$ introduced are immediately "opened" by a $\mathsf{CS}_\Box$ application, hence we know that the side-conditions of $R_\Box$ hold every time we apply R. Therefore we can replace all applications of R by $R_\Box$, which yields:

$$\mathfrak{F}((\mathsf{CS}_\Box+\mathsf{2Box}+R_\Box)^*\cdot\{\mathsf{FA}(b,b')\}^*\cdot{\mathsf{FA_s}}^*\cdot{\mathsf{Dup}}^*\cdot\mathsf{U})$$

Finally we commute the $\mathsf{CS}_\Box$ applications to the right. ∎

B. The Freeze Strategy

We now show that we can restrict the terms on which the rules in $\{\mathsf{FA}(b,b')\}$ can be applied: when we apply a rule in $\{\mathsf{FA}(b,b')\}$, we "freeze" the conditionals b and b' to forbid any further applications of $\{\mathsf{FA}(b,b')\}$ to them.

Example 5. Let $a_i \equiv \mathsf{if}\ b_i\ \mathsf{then}\ c_i\ \mathsf{else}\ d_i$ $(i\in\{1,2\})$, we want to forbid the following partial derivation to appear:

$$\dfrac{\dfrac{b_1,c_1,d_1,u_1,v_1\sim b_2,c_2,d_2,u_2,v_2}{a_1,u_1,v_1\sim a_2,u_2,v_2}\ \mathsf{FA}(b_1,b_2)}{\mathsf{if}\ a_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1\sim\mathsf{if}\ a_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{FA}(a_1,a_2)$$

a) Freeze Strategy: We let $^-$ be a new function symbol of arity one, and for every ground term s we let $\widetilde{s}$ be the term:

$$\widetilde{s}\equiv\begin{cases}\mathsf{if}\ \overline{b}\ \mathsf{then}\ u\ \mathsf{else}\ v & \text{if } s\equiv\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v\\ s & \text{if } s\in\mathcal{T}(\mathcal{F}_s,\mathcal{N})\end{cases}$$

Moreover we replace every $\mathsf{FA}(b_1,b_2)$ rule by the rule:

$$\dfrac{\vec{w}_1,\widetilde{b_1},u_1,v_1\sim\vec{w}_2,\widetilde{b_2},u_2,v_2}{\vec{w}_1,\mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1\sim\vec{w}_2,\mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(b_1,b_2)$$

We let $\{\overline{\mathsf{BFA}}(b_1,b_2)\}$ be the restriction of $\{\mathsf{BFA}(b_1,b_2)\}$ to the rules where b_1 and b_2 are not frozen conditionals. Finally, we add a new rule, UnF, which unfreezes all conditionals: every $\overline{b}$ is replaced by b.

Lemma 7. *The following strategy:*

$$\mathfrak{F}((\mathsf{2Box}+R_\Box)^*\cdot\mathsf{CS}_\Box^*\cdot\{\overline{\mathsf{BFA}}(b,b')\}^*\cdot\mathsf{UnF}\cdot{\mathsf{FA_s}}^*\cdot{\mathsf{Dup}}^*\cdot\mathsf{U})$$

is complete for $\mathfrak{F}((\mathsf{CS}+\mathsf{FA}+R+\mathsf{Dup}+\mathsf{U})^*)$.

Proof. Basically, the proof consists in eliminating all proof cuts of the shape given in Example 5. The cut elimination is simple, though voluminous, and is given in the long version [25]. ∎

VI. Proof Form and Key Properties

The goal of this section is to show that we can assume w.l.o.g. that the terms appearing in the proof (following the ordered freeze strategy) after the $(\mathsf{2Box}+R_\Box)^*$ part have a particular form, that we call proof form. We also show properties of this restricted shape that allow more cut eliminations.

A. Shape of the Terms

Most of the completeness results shown before are for any set of unitary axioms closed under Restr. We now specialize these results to CCA2, to get some further restrictions.

When applying the unitary axioms CCA2, we would like to require that terms are in R-normal form, e.g. to avoid the application of CCA2 to terms with an unbounded component, such as $\pi_1(\langle u\,,\,v\rangle)$. Unfortunately, the side-conditions of CCA2 are not stable under R. E.g., consider the CCA2 instance:

$$\dfrac{}{\{\mathsf{if}\ \mathsf{eq}(g(\mathsf{n}_u),\mathsf{n}_u)\ \mathsf{then}\ A\ \mathsf{else}\ B\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}\sim\{C\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}}\ \mathsf{CCA2}$$

The R-normal form of the left term is:

$$\mathsf{if}\ \mathsf{eq}(g(\mathsf{n}_u),\mathsf{n}_u)\ \mathsf{then}\ \{A\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}\ \mathsf{else}\ \{B\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$$

which cannot be used in a valid CCA2 instance, since the conditional $\mathsf{eq}(g(\mathsf{n}_u),\mathsf{n}_u)$ should be somehow "hidden" by the encryption. To avoid this difficulty, we use a different normal form for terms: we try to be as close as possible to the R-normal form, while keeping conditional branching below their encryption. First, we illustrate this on an example. The term:

$$\{\mathsf{if}\ (\mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c)\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}^{\mathsf{n}_1}_{\mathsf{pk}}\ \mathsf{else}\ w\}^{\mathsf{n}_2}_{\mathsf{pk}}$$

is normalized as follows:

$$\left\{\begin{array}{l}\mathsf{if}\ b\ \mathsf{then}\ \mathsf{if}\ a\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}^{\mathsf{n}_1}_{\mathsf{pk}}\ \mathsf{else}\ w\\ \quad\mathsf{else}\ \mathsf{if}\ c\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}^{\mathsf{n}_1}_{\mathsf{pk}}\ \mathsf{else}\ w\end{array}\right\}^{\mathsf{n}_2}_{\mathsf{pk}}$$

a) Basic Terms: We omit the rewriting strategy here , and describe instead the properties of the normalized terms. We let $\mathcal{A}_\succ$ be the ordered strategy from Lemma 7, and $\mathcal{A}_{\mathsf{CS}_\Box}$ be its restriction to proofs with an empty $(\mathsf{2Box}+R_\Box)^*$ part. The rule $\mathsf{CS}_\Box$ is the only branching rule, therefore, after applying all the $\mathsf{CS}_\Box$ rules, we can associate to each branch l of the proof an instance $\mathcal{S}_l=(\mathcal{K}_l,\mathcal{R}_l,\mathcal{E}_l,\mathcal{D}_l)$ of the CCA2 axiom, where $\mathcal{K}_l$, $\mathcal{R}_l$, $\mathcal{E}_l$ and $\mathcal{D}_l$ are the sets of, respectively, secret keys, encryption randomness, encryptions and decryptions. We use $\mathcal{S}_l$ to define a normal form for the terms appearing in branch l. This is done through four mutually inductive definitions: *$\mathcal{S}_l$-encryption oracle calls* are well-formed encryptions; *$\mathcal{S}_l$-decryption oracle calls* are well-formed decryptions; *$\mathcal{S}_l$-normalized basic terms* are terms built using function symbols in $\mathcal{F}_s$ and well-formed encryptions and decryptions; and *$\mathcal{S}_l$-normalized simple terms* are combinations

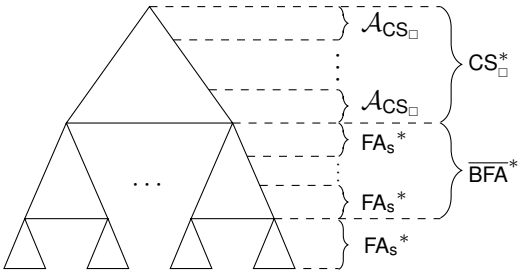


Fig. 4. The shape of the term is determined by the proof.

of normalized basic terms using if_then_else_. We give only the definition of $\mathcal{S}_l$-normalized basic terms (the full definitions are in the long version [25]).

Definition 6. *A $\mathcal{S}_l$-*normalized basic term *is a term t of the form $U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$ where:*

- *U and $\vec{w}$ are if-free and $\mathcal{R}_l, \mathcal{K}_l$ do not appear in $\vec{w}$.*
- *$U[\vec{w}, (\{[]_j\}^{n_j}_{\mathsf{pk}_j})_j, (\mathsf{dec}([]_k, \mathsf{sk}_k))_k]$ is in R-normal form.*
- *$(\alpha_j)_j$ are $\mathcal{S}_l$-*encryption oracle calls *under $(\mathsf{pk}_j, \mathsf{sk}_j)_j$.*
- *$(\mathsf{dec}_k)_k$ are $\mathcal{S}_l$-*decryption oracle calls *under $(\mathsf{pk}_k, \mathsf{sk}_k)_k$.*

*If t is of sort bool, we say that it is a $\mathcal{S}_l$-*normalized basic conditional.

b) Normalized Proof Form: Every application of $\mathsf{CS}_\Box$:

$$\frac{a_1, u \sim b_1, s \qquad a_2, v \sim b_2, t}{\mathsf{if}\ \boxed{a_1}\boxed{a_2}_a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ \boxed{b_1}\boxed{b_2}_b\ \mathsf{then}\ s\ \mathsf{else}\ t}\ \mathsf{CS}_\Box$$

is such that if we extract the sub-proof of $a_i \sim b_i$ (for $i \in \{1, 2\}$), we get a proof in $\mathcal{A}_{\mathsf{CS}_\Box}$. Therefore, we can check that terms after $(\mathsf{2Box} + R_\Box)^*$ are of the form informally described in Fig. 4. We define a normal form for such proofs, called *normalized proof form*, and we define $\vdash^{\mathsf{npf}}$ by $P \vdash^{\mathsf{npf}} t \sim t'$ if and only if $P \vdash t \sim t'$, the proof P is in $\mathcal{A}_{\succ}$ and is in *normalized proof form*. We do not give the full definition, but one of the key ingredients is to require that for every term s appearing in a branch l of the proof P, if s is the conclusion of a sub-proof in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$ then s is a $\mathcal{S}_l$-normalized basic term.

Lemma 8. *Every formula in $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{CCA2})^*)$ is provable using the strategy $\vdash^{npf}$.*

Proof. (*sketch*) The full proof is in the long version [25]. First, we rewrite terms by pulling conditionals upward without crossing an encryption function symbol, and without modifying decryption guards. Then, we remove all redexes from R_1 (e.g. $\pi_1(\langle u\,,\, v\rangle) \to u$) using a cut elimination procedure. E.g., the following cut can be eliminated using Lemma 1:

$$\frac{\dfrac{u, v \sim u', v'}{\pi_1(\langle u\,,\, v\rangle) \sim \pi_1(\langle u'\,,\, v'\rangle)}\ \mathsf{FA}_{\langle\,,\,\rangle}}{u \sim u'}\ R$$

■

B. Key Properties

A term in R-normal form is in the following grammar:

$$t ::= u \in \mathcal{T}(\mathcal{F}_s, \mathcal{N}) \mid \mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ t\ \ (\text{with } b \in \mathcal{T}(\mathcal{F}_s, \mathcal{N}))$$

Given a term t in R-normal form, we let $\mathsf{cond\text{-}st}(t)$ be its set of conditionals, and $\mathsf{leave\text{-}st}(t)$ its set of leaves.

a) Characterization of Basic Terms: We give a key characterization proposition for basic terms: if two $\mathcal{S}_l$-normalized basic terms β and β' are such that, when R-normalizing them, they share a leaf term, then they are identical.

Proposition 5. *For all $\mathcal{S}_l$-normalized basic terms β, β', if we have $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta' \downarrow_R) \neq \emptyset$ then $\beta \equiv \beta'$.*

Proof. (*sketch*) The full proof is in the long version [25]. We give the intuition: since they are $\mathcal{S}_l$-normalized basic terms, we know that $\beta \equiv U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$, $\beta' \equiv U'[\vec{w}', (\alpha'_j)_j, (\mathsf{dec}'_k)_k]$ and:

$$U[\vec{w}, (\{[]_j\}^{\mathsf{n}_j}_{\mathsf{pk}_j})_j, (\mathsf{dec}([]_k, \mathsf{sk}_k))_k]$$
$$U'[\vec{w}', (\{[]'_j\}^{\mathsf{n}_j}_{\mathsf{pk}'_j})_j, (\mathsf{dec}([]'_k, \mathsf{sk}'_k))_k]$$

are in R-normal form. Using the fact that $U, U', \vec{w}, \vec{w}'$ are if-free, and the hypothesis that β and β' share a leaf term, we first show that we can assume $U \equiv U'$ and $\vec{w} \equiv \vec{w}'$ by induction on the number of positions where U and U' differ. Take p where they differ, w.l.o.g. assume $\beta'_{|p}$ to be a hole of U' (otherwise swap β and β'). We have three cases: i) if $\beta_{|p}$ is in $\vec{w}$, we simply change U to include everything up to p; ii) if $\beta_{|p}$ is in some encryption $\alpha_j \equiv \{m\}^{\mathsf{n}}_{\mathsf{pk}}$, then we know that n appears in $\vec{w}$, which is not possible since, as β is a $\mathcal{S}_l$-normalized basic term, $\mathsf{n} \in \mathcal{R}_l$ does not appear in $\vec{w}$; iii) if $\beta_{|p}$ is in some decryption $\mathsf{dec}_k \equiv \mathsf{dec}(u_k, \mathsf{sk}_k)$ then, similarly to the previous case, we have sk_k appearing in $\vec{w}$, which contradicts the fact that $\mathsf{sk}_k \in \mathcal{K}_l$ do not appear in $\vec{w}$.

Knowing that $U \equiv U'$ and $\vec{w} \equiv \vec{w}'$, it only remains to show that the encryptions $(\alpha_j)_j$ and $(\alpha'_j)_j$, and the decryptions $(\mathsf{dec}_k)_k$ and $(\mathsf{dec}'_k)_k$ are identical. The former follows from the fact that, for a given encryption randomness $\mathsf{n} \in \mathcal{R}_l$, there exists a unique m such $\{m\}^{\mathsf{n}}_{\_} \in \mathcal{E}_l$; and the latter follows from the fact that there is a unique way to guard a decryption in $\mathcal{D}_l$ (this is not obvious, and relies on CCA2 side-conditions). ■

b) Proofs of $b \sim$ false or true: Using the previous proposition, we can show that for all b, if b is if-free then there is no derivation of $b \sim \mathsf{true}$ or $b \sim \mathsf{false}$ in $\mathcal{A}_{\succ}$. Such derivations would be problematic since true and false are conditionals of constant size, but b could be of any size (and we are trying to bound all conditionals appearing in a proof). Also, the else branch of a true conditional can contain anything and is, a priori, not bounded by the proof conclusion.

Proposition 6. *Let b an if-free conditional in R-normal form, with $b \not\equiv$ false (resp. $b \not\equiv$ true). Then there exists no derivation of $b \sim$ false (resp. $b \sim$ true) in $\mathcal{A}_{\succ}$.*

Proof. This is shown by induction on the size of the derivation. The full proof is in the long version [25], and relies on Proposition 5. ■

VII. Bounding the Proof and Decision Procedure

We give here two similar proof cut eliminations, one used on $\overline{\mathsf{BFA}}$ conditionals and the other on $\mathsf{CS}_\Box$ conditionals.

a) $\overline{\mathsf{BFA}}$ Rule: We already used this cut elimination to deal with Example 4 for conditionals involved in $\overline{\mathsf{BFA}}$ applications. The cuts we want to eliminate are of the form:

$$\dfrac{a_1, a_2, u_3, v_4, w_5 \sim b_1, c_2, r_3, s_4, t_5}{\begin{array}{ccc} a_1 & & b_1 \\ a_2 \quad w_5 & \sim & c_2 \quad t_5 \\ u_3 \quad v_4 & & r_3 \quad s_4 \\ \underbrace{\qquad}_{\sigma} & & \underbrace{\qquad}_{\tau} \end{array}}\ \overline{\mathsf{BFA}}^{(2)} \tag{4}$$

Using Lemma 1, we extract a proof of $a_1, a_2 \sim b_1, c_2$, which, thanks to the ordered strategy, is in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$. From Lemma 2 we get that $b \equiv c$. We then replace (4) by:

$$\dfrac{a_1, u_3, w_5 \sim b_1, r_3, t_5}{\begin{array}{ccc} a_1 & & b_1 \\ u_3 \quad w_5 & \sim & r_3 \quad t_5 \\ & \sigma \sim \tau & \end{array}}\ \begin{array}{l} \overline{\mathsf{BFA}} \\ R \end{array}$$

We retrieve a proof in $\mathcal{A}_{\succ}$ by pulling R to the beginning of the proof.

b) $\mathsf{CS}_\Box$ Rule: The $\mathsf{CS}_\Box$ case is more complicated. E.g., take two boxed $\mathsf{CS}_\Box$ conditionals for the same if-free conditional a, and two arbitrary $\mathsf{CS}_\Box$ conditionals on the right side:

$$a_i^\Box \equiv \boxed{a_i^l \mid a_i^r}_a \ (i \in \{1,2\}) \quad b_1^\Box \equiv \boxed{b_1^l \mid b_1^r}_b \quad c_2^\Box \equiv \boxed{c_2^l \mid c_2^r}_c$$

Consider the following cut:

$$\dfrac{\begin{array}{ccc} \vdots\ (A) & \vdots\ (B) & \vdots\ (C) \\ a_1^l, a_2^l, u_3 \sim b_1^l, c_2^l, r_3 & a_1^l, a_2^r, v_4 \sim b_1^l, c_2^r, s_4 & a_1^r, w_5 \sim b_1^r, t_5 \end{array}}{\begin{array}{ccc} a_1^\Box & & b_1^\Box \\ a_2^\Box \quad w_5 & \sim & c_2^\Box \quad t_5 \\ u_3 \quad v_4 & & r_3 \quad s_4 \\ \underbrace{\qquad}_{\sigma} & & \underbrace{\qquad}_{\tau} \end{array}}\ \mathsf{CS}_\Box^{(2)}$$

As we did for $\overline{\mathsf{BFA}}$, we can extract from (A), using Lemma 1, a proof of $a_1^l, a_2^l \sim b_1^l, c_2^l$. But using the ordered strategy, we get that this proof is in $\mathcal{A}_{\mathsf{CS}_\Box}$, which we recall is the fragment:

$$\mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$$

Therefore we cannot apply Lemma 2. To deal with this cut, we generalize Lemma 2 to the case where the proof is in $\mathcal{A}_{\mathsf{CS}_\Box}$. For this, we need the extra assumptions that $a_1^l, a_2^l, b_1^l, c_2^l$ are if-free, which is a side-condition of $\mathsf{CS}_\Box$.

Lemma 9. *For all a, a', b, c such that their R-normal form is if-free and $a =_R a'$, if $P \vdash^{npf} a, a' \sim b, c$ then $b =_R c$.*

Proof. (*sketch*) The full proof is given in the long version [25]. It uses Proposition 6 to obtain a proof P' of $a, a' \sim b, c$ without any false and true, and also relies on Proposition 5 and Lemma 2. ∎

We now deal with the cut above. Using Lemma 9, we know that $b =_R c$. Since b, c are in R-normal form, $b \equiv c$ and therefore $b_1^\Box =_{R_\Box} b =_{R_\Box} c_2^\Box$ (using well-formedness). Similarly $a_1^\Box =_{R_\Box} a =_{R_\Box} a_2^\Box$. This yields the (cut-free) proof:

$$\dfrac{\begin{array}{cc} \vdots\ (A') & \vdots\ (C) \\ a_1^l, u_3 \sim b_1^l, r_3 & a_1^r, w_5 \sim b_1^r, t_5 \end{array}}{\begin{array}{ccc} a_1^\Box & & b_1^\Box \\ u_3 \quad w_5 & \sim & r_3 \quad t_5 \\ & \sigma \sim \tau & \end{array}}\ \begin{array}{l} \mathsf{CS}_\Box \\ R_\Box \end{array}$$

where (A') is extracted from (A) by Lemma 1. Finally, to get a proof in $\mathcal{A}_{\succ}$, we commute the $R_\Box$ rewriting to the beginning.

A. Decision Procedure

Now, we explain how we obtain a decision procedure for our logic. Because the proofs and definitions are long and technical, we omit most of the details and focus instead on giving a high level sketch of the proof and decision procedure.

a) Spurious Conditionals: A conditional b without $\mathsf{if\_then\_else\_}$ and in R-normal form is said to be *spurious* in t if, when R-normalizing t, the conditional b disappears. Formally, b is spurious in t if $b \notin \mathsf{cond\text{-}st}(t \downarrow_R)$. E.g., the conditional $\mathsf{eq}(\mathsf{n}_0, \mathsf{n}_1)$ is spurious in:

$$\mathsf{if}\ \mathsf{eq}(\mathsf{n}_0, \mathsf{n}_1)\ \mathsf{then}\ g(\mathsf{n})\ \mathsf{else}\ g(\mathsf{n})$$

We say that a basic conditional β, which may not be if-free, is spurious in t if all its leaf terms are spurious in t (i.e. $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{cond\text{-}st}(t \downarrow_R) = \emptyset$). As we saw in Example 2, we may need to introduce spurious basic conditionals to carry out a proof. Still, we need to bound such terms. To do this, we characterize the basic conditionals that *cannot* be removed: basically, a basic conditional is α-bounded in a proof of $t \sim t'$ if it is not spurious in t or t', or if it is a guard for a decryption appearing in a α-bounded conditional of $t \sim t'$ (indeed, we cannot remove a decryption's guards, as this would not yield a valid $\mathsf{CCA2}$ instance).

We let $\vdash^{\mathsf{npf}}_{\alpha}$ be the restriction of $\vdash^{\mathsf{npf}}$ to proofs such that all basic conditionals appearing in the derivation are α-bounded. Using the cut eliminations we introduced earlier, plus some additional cut eliminations, we can show the following completeness result (the full proof is in the long version [25]).

Lemma 10. *$\vdash^{npf}_{\alpha}$ is complete with respect to $\vdash^{npf}$.*

b) Bounding α-bounded Basic Conditionals: Finally, it remains to bound the size of α-bounded basic conditionals. Since basic conditionals can be nested (e.g. a basic conditional can contain decryption guards, which are themselves basic conditionals etc), we need to bound the length of sequences of nested basic conditionals.

Given a sequence of nested basic conditionals $\beta_1 <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_n$, (where $u <_{\mathsf{st}} v$ iff $u \not\equiv v$ and $u \in \mathsf{st}(v)$), we show that we can associate to each β_i a "frame term" $\lambda_i \in \mathcal{B}(t, t')$ (where $\mathcal{B}(t, t')$ is a set of terms of bounded size w.r.t. $|t| + |t'|$). Basically, λ_i is obtained from β_i by

"flattening" it: we remove all decryption guards, and replace the content of every encryption $\{m\}^{\mathsf{n}}_{\mathsf{pk}}$ by a term $\{\tilde{m}\}^{\mathsf{n}}_{\mathsf{pk}}$, where $\tilde{m}$ is if-free and in $\mathcal{B}(t,t')$. Moreover, we show that, for every $\mathcal{S}_l$-normalized basic terms β,γ and their associated frame terms λ,μ, if $\lambda \equiv \mu$ then $\beta \equiv \gamma$ (this result is similar to Proposition 5).

Since the β_is are all pair-wise distinct (as $<_{\mathsf{st}}$ is strict), and since for every i, the frame term λ_i uniquely characterizes β_i, we know that the λ_is are pair-wise distinct. Using a pigeon-hole argument, this shows that $n \leq |\mathcal{B}(t,t')|$. Then, by induction on the number of nested basic conditionals, we show a triple exponential upper-bound in $|t|+|t'|$ on the size of the basic conditionals appearing in a cut-free proof of $t \sim t'$.

c) Decision Procedure: To conclude, we show that there exists a non-deterministic procedure that, given two terms t and t', non-deterministically guesses a set of α-bounded basic terms that can appear in a proof P of $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$ (in triple exponential time in $|t|+|t'|$). Then the procedure guesses the rule applications, and checks that the candidate derivation is a valid proof (in polynomial time in the candidate derivation size). This yields a 3-NEXPTIME decision procedure that shows the decidability of our problem.

Theorem (Main Result)**.** *The following problem is decidable:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?*

VIII. RELATED WORKS

In [29], the authors design a set of inference rules to prove CPA and CCA security of asymmetric encryption schemes in the Random Oracle Model. The paper also presents an attack finding algorithm. The authors of [29] do not provide decision algorithm for the designed inference rules. However, they designed proof search heuristics and implemented an automated tool, called ZooCrypt, to synthesize new CCA encryption schemes. For small schemes, this procedure can show CCA security or find an attack in more than 80% of the cases. In 20% of the cases, security remains undecided. Additionally, ZooCrypt automatically generates concrete security bounds.

As seen in the introduction, the problem of showing CPA security can be cast into the BC logic. Take a candidate encryption scheme $x \mapsto t[x]$, where $t[]$ is a context built using, e.g., pairs, a one-way permutation f using public key $\mathsf{pk}(\mathsf{n})$, hash functions and xor. Then this scheme is CPA if the following formula is valid in every computational model satisfying some implementation assumptions (mostly, f is OW-CPA and the hash functions are PRF):

$$t[\pi_1(f(\mathsf{pk}(\mathsf{n})))] \sim t[\pi_2(f(\mathsf{pk}(\mathsf{n})))]$$

This formula has a particular shape, which stems from the limitations on the adversary's interactions: the adversary can only interact with the (candidate) encryption scheme through the CPA or CCA game. There is no complex and arbitrary interactions with the adversary, as it is the case with a security protocol. We don't have such restrictions.

In [30], the authors study proof automation in the UC framework [31]. They design a complete procedure for deciding the existence of a simulator, for ideal and real functionalities using if-then-else, equality, random samplings and xor. Therefore their algorithm cannot be used to analyse functionalities relying on more complex functions (e.g., public key encryption), or stateful functionalities. This restricts the protocols that can be checked. Still, their method is *semantically* complete (while we are complete w.r.t. a fixed set of inference rules): if there exists a simulator, they will find it.

In [32], the authors show the decidability of the problem of the equality of two distributions, for a *specific* equational theory (concatenation, projection and xor). Then, for *arbitrary* equational theories, they design a proof system for proving the equality of two distributions. This second contribution has similarities with our work, but differ in two ways.

First, the proof system of [32] shares some rules with ours, e.g. the R, Dup and FA rules. But it does not allow for reasoning on terms using if_then_else_. E.g., they do not have a counterpart to the CS rule. This is a major difference, as most of the difficulties encountered in the design of our decision procedure result from the if_then_else_ conditionals. Moreover, there are no rules corresponding to cryptographic assumptions, as our CCA2 rules. Because of this and the lack of support for reasoning on branching terms, the analysis of security protocols is out of the scope of [32].

Second, the authors do not provide a decision procedure for their inference rules, but instead rely on heuristics.

IX. CONCLUSION

We designed a decision procedure for the Bana-Comon indistinguishability logic. This allows to automatically verify that a security protocol satisfies some security property. Our result can be reinterpreted, in the cryptographic game transformation setting, as a cut elimination procedure that guarantees that all intermediate games introduced in a proof are of bounded size w.r.t. the protocol studied.

A lot of work remains to be done. First, our decision procedure is in 3-NEXPTIME, which is a high complexity. But, as we do not have any lower-bound, there may exist a more efficient decision procedure. Finding such a lower-bound is another interesting direction of research. Then, our completeness result was proven for CCA2 only. We believe it can be extended to more primitives and cryptographic assumptions. For example, signatures and EUF-CMA are very similar to asymmetric encryption and IND-CCA$_2$, and should be easy to handle (even combined with the CCA2 axioms).

ACKNOWLEDGMENT

We thank Hubert Comon for his help and useful comments.

This research has been partially funded by the French National Research Agency (ANR) under the project TECAP (ANR-17-CE39-0004-01).

REFERENCES

[1] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *2014 ACM Conference on Computer and Communications Security, CCS '14*. ACM, 2014, pp. 609–620.

[2] G. Bana and H. Comon-Lundh, "Towards unconditional soundness: Computationally Complete Symbolic Attacker," in *Principles of Security and Trust, 2012*, ser. LNCS, vol. 7215. Springer, 2012, pp. 189–208.

[3] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Z. Béguelin, and P. Zimmermann, "Imperfect forward secrecy: How Diffie-Hellman fails in practice," in *ACM Conference on Computer and Communications Security*. ACM, 2015, pp. 5–17.

[4] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, and P. Strub, "Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2014, pp. 98–113.

[5] B. Blanchet, PROVERIF*: Cryptographic protocols verifier in the formal model*, available at http://proseccco.gforge..inria.fr/personal/bblanchet/proverif/.

[6] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *25th International Conference on Computer Aided Verification, CAV'13*. Springer-Verlag, 2013, pp. 696–701.

[7] V. Cheval, S. Kremer, and I. Rakotonirina, "DEEPSEC: deciding equivalence properties in security protocols theory and practice," in *2018 IEEE Symposium on Security and Privacy, SP 2018*. IEEE, 2018, pp. 529–546.

[8] G. Barthe, B. Grégoire, S. Heraud, and S. Z. Béguelin, "Computer-aided security proofs for the working cryptographer," in *Advances in Cryptology - CRYPTO, 2011*, ser. LNCS, vol. 6841. Springer, 2011, pp. 71–90.

[9] B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Sec. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.

[10] H. Comon-Lundh, V. Cortier, and E. Zalinescu, "Deciding security properties for cryptographic protocols. application to key cycles," *ACM Trans. Comput. Log.*, vol. 11, no. 2, pp. 9:1–9:42, 2010.

[11] E. D'Osualdo, L. Ong, and A. Tiu, "Deciding secrecy of security protocols for an unbounded number of sessions: The case of depth-bounded processes," in *CSF*. IEEE Computer Society, 2017, pp. 464–480.

[12] A. Finkel and P. Schnoebelen, "Well-structured transition systems everywhere!" *Theor. Comput. Sci.*, vol. 256, no. 1-2, pp. 63–92, 2001.

[13] R. Chrétien, V. Cortier, and S. Delaune, "Decidability of trace equivalence for protocols with nonces," in *CSF*. IEEE Computer Society, 2015, pp. 170–184.

[14] V. Cheval, H. Comon-Lundh, and S. Delaune, "A procedure for deciding symbolic equivalence between sets of constraint systems," *Inf. Comput.*, vol. 255, pp. 94–125, 2017.

[15] H. Comon-Lundh, V. Cortier, and G. Scerri, "Tractable inference systems: An extension with a deducibility predicate," in *CADE*, ser. LNCS, vol. 7898. Springer, 2013, pp. 91–108.

[16] M. Abadi and P. Rogaway, "Reconciling two views of cryptography (the computational soundness of formal encryption)," *J. Cryptology*, vol. 15, no. 2, pp. 103–127, 2002.

[17] M. Backes, A. Malik, and D. Unruh, "Computational soundness without protocol restrictions," in *ACM Conference on Computer and Communications Security*. ACM, 2012, pp. 699–711.

[18] M. Backes, E. Mohammadi, and T. Ruffing, "Computational soundness results for proverif - bridging the gap from trace properties to uniformity," in *POST*, ser. Lecture Notes in Computer Science, vol. 8414. Springer, 2014, pp. 42–62.

[19] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *CRYPTO*, ser. LNCS, vol. 1462. Springer, 1998, pp. 26–45.

[20] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," *IACR Cryptology ePrint Archive*, vol. 2004, p. 332, 2004, https://eprint.iacr.org/2004/332.

[21] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *EUROCRYPT*, ser. LNCS, vol. 4004. Springer, 2006, pp. 409–426.

[22] H. Comon and A. Koutsos, "Formal computational unlinkability proofs of RFID protocols," in *30th Computer Security Foundations Symposium, 2017*. IEEE Computer Society, 2017, pp. 100–114.

[23] G. Scerri and R. Stanley-Oakes, "Analysis of key wrapping APIs: Generic policies, computational security," in *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*. IEEE Computer Society, 2016, pp. 281–295.

[24] G. Bana, R. Chadha, and A. K. Eeralla, "Formal analysis of vote privacy using computationally complete symbolic attacker," in *ESORICS (2)*, ser. LNCS, vol. 11099. Springer, 2018, pp. 350–372.

[25] A. Koutsos, "Deciding indistinguishability," *CoRR*, vol. abs/1811.06936, 2018.

[26] C. Chang and R. C. T. Lee, *Symbolic logic and mechanical theorem proving*, ser. Computer science classics. Academic Press, 1973.

[27] G. Bana and R. Chadha, "Verification methods for the computationally complete symbolic attacker based on indistinguishability," *IACR Cryptology ePrint Archive*, vol. 2016, p. 69, 2016. [Online]. Available: http://eprint.iacr.org/2016/069

[28] G. Lowe, "An attack on the Needham-Schroeder public-key authentication protocol," *Inf. Process. Lett.*, vol. 56, no. 3, pp. 131–133, 1995.

[29] G. Barthe, J. M. Crespo, B. Grégoire, C. Kunz, Y. Lakhnech, B. Schmidt, and S. Z. Béguelin, "Fully automated analysis of padding-based encryption in the computational model," in *ACM Conference on Computer and Communications Security*. ACM, 2013, pp. 1247–1260.

[30] C. S. Jutla and A. Roy, "Decision procedures for simulatability," in *ESORICS*, ser. LNCS, vol. 7459. Springer, 2012, pp. 573–590.

[31] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *FOCS*. IEEE Computer Society, 2001, pp. 136–145.

[32] G. Barthe, M. Daubignard, B. M. Kapron, Y. Lakhnech, and V. Laporte, "On the equality of probabilistic terms," in *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers*, ser. LNCS, E. M. Clarke and A. Voronkov, Eds., vol. 6355. Springer, 2010, pp. 46–63.