Proof of security protocols revisited

Guillaume Scerri

EDSP
École Doctorale
Sciences Pratiques

ENSC-2015-nº 561

ENS
CACHAN

Thèse de doctorat
de l'École Normale Supérieure de Cachan

Présentée par

Monsieur Guillaume Scerri

Pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

Domaine :
Informatique

Sujet de la thèse :

Les preuves de protocoles cryptographiques revisitées

(Proofs of security protocols revisited)

Thèse présentée et soutenue à Cachan le 29 janvier 2015 devant le jury composé de :

Gilles Barthe	Professeur	Président du Jury
David Basin	Professeur	Rapporteur
Bruno Blanchet	Directeur de Recherches	Rapporteur
Hubert Comon-Lundh	Professeur	Co-directeur de thèse
Véronique Cortier	Directrice de Recherches	Co-directrice de thèse
Pierre-Alain Fouque	Professeur	Examinateur

Laboratoire Spécification et Vérification
ENS de Cachan, UMR 8643 du CNRS
61, avenue du Président Wilson
94235 CACHAN Cedex, France

Abstract

With the rise of the Internet the use of cryptographic protocols became ubiquitous. Considering the criticality and complexity of these protocols, there is an important need of formal verification.

In order to obtain formal proofs of cryptographic protocols, two main attacker models have been developed: the symbolic model and the computational model. The symbolic model defines the attacker capabilities as a fixed set of rules. On the other hand, the computational model describes only the attacker's limitations by stating that it may break some hard problems. While the former is quite abstract and convenient for automating proofs the later offers much stronger guarantees.

There is a gap between the guarantees offered by these two models due to the fact the symbolic model defines what the adversary may do while the computational model describes what it may not do. Since Abadi and Rogaway in 2000 a lot of results aimed at bridging this gap, in order to have symbolic proofs yielding computational guarantees. They however all come at the cost of very strong and often unrealistic hypotheses, due to the fact that attacker capabilities are defined in a fundamentally different way in the two models.

In order to overcome this problem, in 2012 Bana and Comon devised a new symbolic model in which the attacker's limitations are axiomatised. Proving security in this new model amounts to checking – for every trace of the protocol – the unsatisfiability of a set of formulae corresponding to the trace, together with the negation of the security property and some axioms representing the attacker's limitations. In addition, provided that the (computational semantics) of the axioms follows from the cryptographic hypotheses, proving security in this symbolic model yields security in the computational model.

The possibility of automating proofs in this model (and finding axioms general enough to prove a large class of protocols) was left open in the original paper from Bana and Comon. In this thesis we provide with an efficient decision procedure for a general class of axioms. In addition we propose a tool (SCARY) implementing this decision procedure. Experimental results of our tool shows that the axioms we designed for modelling security of encryption are general enough to prove a large class of protocols.

Résumé

Avec la généralisation d'Internet, l'usage des protocoles cryptographiques est devenu omniprésent. Étant donné leur complexité et leur aspect critique, une vérification formelle des protocoles cryptographiques est nécessaire.

Deux principaux modèles existent pour prouver les protocoles. Le modèle symbolique définit les capacités de l'attaquant comme un ensemble fixe de règles, tandis que le modèle calculatoire interdit seulement à l'attaquant de résoudre certains problèmes difficiles. Le modèle symbolique est très abstrait et permet généralement d'automatiser les preuves, tandis que le modèle calculatoire fournit des garanties plus fortes.

Le fossé entre les garanties offertes par ces deux modèles est dû au fait que le modèle symbolique décrit les capacités de l'adversaire alors que le modèle calculatoire décrit ses limitations. Depuis l'article précurseur d'Abadi et Rogaway en 2000, de nombreux travaux ont tenté de combler ce fossé. Ils supposent tous des hypothèses très fortes et souvent irréalistes sur l'implémentation des protocoles. La nécessité de ces hypothèses s'explique par le fait que les modèles d'attaquant sont fondamentalement différents dans le monde calculatoire et dans le monde symbolique. En effet, dans le modèle calculatoire les capacités de l'attaquant sont un plus grand point fixe alors qu'elles sont un plus petit point fixe dans le monde symbolique. Ainsi tout résultat voulant obtenir des garanties calculatoires à partir de preuves de sécurité symbolique doit s'assurer que ces deux points fixes coïncident.

Pour pallier ce problème, en 2012 Bana et Comon ont proposé un nouveau modèle symbolique dans lequel les limitations de l'attaquant (au lieu de ses capacités) sont axiomatisées. En pratique, prouver la sécurité d'un protocole dans ce modèle revient à montrer – pour chaque trace du protocole – que les axiomes limitant l'attaquant ainsi que les formules correspondant à la trace sont inconsistants avec la négation de la propriété de sécurité. Les auteurs montrent que si la sémantique calculatoire des axiomes découle des hypothèses cryptographiques, la sécurité dans ce modèle symbolique fournit des garanties calculatoires.

La principale question laissée ouverte par l'article de Bana et Comon est la possibilité d'automatiser les preuves dans ce nouveau modèle (et l'élaboration d'axiomes suffisamment généraux pour prouver un grand nombre de protocoles). Le principal problème à résoudre pour automatiser les preuves dans ce nouveau modèle est la résolution de problèmes de satisfiabilité d'ensembles de formules du premier ordre. Dans cette thèse nous identifions un sous-ensemble de la logique du premier ordre englobant les ensembles de formules et axiomes considérés dans ce modèle, et proposons une procédure de décision originale en temps polynomial pour cette classe.

Dans un deuxième temps, nous proposons une implémentation de cette procédure de décision dans notre outil (SCARY). Nos résultats expérimentaux nous ont dans un premier temps amené à élargir les axiomes modélisant la sécurité du chiffrement. Nous pensons que les concepts utilisés pour étendre ces axiomes pourront être réutilisés dans l'élaboration d'axiomes pour d'autres primitives. Dans un deuxième temps nous avons montré que nos nouveaux axiomes sont suffisamment généraux pour prouver une large classe de protocoles. Nous avons également retrouvé des attaques connues sur divers protocoles qui ne sont pas détectées par les outils symboliques usuels. Nous avons, enfin, trouvé une attaque originale sur le protocole Andrews Secure RPC.

Remerciements

En premier lieu, je tiens à remercier mes deux directeurs de thèse : Hubert Comon et Véronique Cortier. Au delà de m'avoir fait découvrir ce beau domaine de recherche qu'est la vérification de protocoles cryptographiques et d'avoir toujours été disponibles pour discuter de nos idées et orientations de recherche, ils m'ont toujours, durant ces trois ans, soutenu et poussé à continuer et aller plus loin, même dans les périodes de découragement. Pour tout ceci ils ont ma plus profonde gratitude.

Je tiens également à remercier mes deux rapporteurs David Basin et Bruno Blanchet, pour avoir accepté de relire cette thèse et pour leurs nombreux commentaires qui m'ont permis d'en améliorer la qualité. Merci également à Gilles Barthe et Pierre-Alain Fouque pour avoir accepté de participer à mon jury.

Ce travail n'aurait sans doute pas été possible sans les nombreuses discussions scientifiques que j'ai pu avoir avec d'autres chercheurs du LSV et d'ailleurs. Je suis en particulier reconnaissant à Gergeï Bana, pour les nombreux échanges que nous avons eus au cours de ces trois ans, ainsi que pour ses commentaires sur les versions préliminaires de mon manuscrit.

Merci à tous les personnels administratifs et techniques du LSV, du LORIA, et de l'EDSP qui m'ont fourni des conditions matérielles idéales pour cette thèse, toujours avec le sourire.

Je remercie également tous ceux avec qui j'ai collaboré pour mes enseignements, Hubert Comon, Émile Contal, Paul Gastin, Jean Goubault, Malika Izabachène, et Ocan Sankur. Grâce à eux enseigner a toujours été une joie durant ces trois ans. Merci aussi à mes élèves qui ont amplement contribué au plaisir que j'ai pris à enseigner.

Merci à tous ceux avec qui nous avons partagé de bons moments au quotidien au LSV et au LORIA. En particulier, David, Cyrille, Lucca, Marie, Marie, Mathilde, Nadime, Pierre-Arnaud, Rémy, Samy, Simon, Steve, Sylvain. Un grand merci supplémentaire à ceux parmi eux qui ont eu la gentillesse de relire des parties de ma thèse.

Ces remerciements ne seraient évidemment pas complets sans les amis hors du laboratoire (Andréa, Arthur, Aurel, Blanche, Catherine, Florence, Lucas, Néphéli, Nicolas, Robert, Samuel, Sarah, Umberto, Xavier, et tous ceux que j'oublie), avec qui nous avons partagé de nombreux bons moments parfois absurdes, et probablement beaucoup trop joué aux cartes et jeux de société.

Merci à René et Chantal pour m'avoir supporté, et conseillé. Un immense merci à mes parents qui, comme toujours, m'ont inconditionnellement soutenu et aidé durant ces trois ans, qui ont parfois été difficiles. Sans eux je n'aurais probablement pas fini cette thèse.

Merci de tout mon coeur à Maëlle.

Contents

1 Introduction **9**
- 1.1 Some context . . . 9
- 1.2 State of the art . . . 10
  - 1.2.1 The symbolic models . . . 11
  - 1.2.2 The computational model . . . 12
- 1.3 Comparison and critic of existing models . . . 13
  - 1.3.1 Symbolic models vs. computational model . . . 13
  - 1.3.2 Computational soundness . . . 15
  - 1.3.3 The computationally complete symbolic attacker . . . 16
  - 1.3.4 Other symbolic models for a computational attacker . . . 17
- 1.4 Our contribution . . . 18

2 Formal setting **19**
- 2.1 First order semantics . . . 20
  - 2.1.1 Constraints . . . 21
  - 2.1.2 Semantics . . . 22
- 2.2 Kripke semantics . . . 23
  - 2.2.1 Samplings, functions and constants . . . 23
- 2.3 The execution model . . . 27
- 2.4 Conclusion . . . 33

3 Axioms and soundness proofs **35**
- 3.1 Core axioms . . . 35
  - 3.1.1 Reflexivity, transitivity and monotonicity . . . 36
  - 3.1.2 Functionality . . . 38
  - 3.1.3 Freshness . . . 38
- 3.2 Cryptographic axioms . . . 40
  - 3.2.1 Cryptographic properties . . . 40
  - 3.2.2 Syntactic constraints for key usability . . . 44
  - 3.2.3 Simple secrecy axiom(s) . . . 47
  - 3.2.4 Key usability and full secrecy axioms . . . 52
  - 3.2.5 Integrity and non-malleability axiom(s) . . . 55
- 3.3 Conclusion . . . 60

4 Decision procedure **61**
- 4.1 Overview of the results and proofs . . . 62
- 4.2 Tractability of deducibility axioms . . . 64
  - 4.2.1 Adding equality . . . 67

4.2.2 Adding a function axiom . . . 68
4.3 More clauses using the deducibility predicate . . . 71
4.3.1 Adding other predicate symbols . . . 76
4.3.2 Adding equality . . . 77
4.4 The general case . . . 79
4.4.1 Proof of Theorem 6 . . . 80
4.5 Conclusion . . . 87

5 SCARY **89**
5.1 Overview of the tool . . . 89
5.2 Input syntax and trace computation . . . 90
5.3 The logic . . . 93
5.4 Computing on terms . . . 95
5.4.1 Equational theory . . . 95
5.4.2 Constraints . . . 96
5.5 Preprocessing and optimisations . . . 99
5.5.1 Functionality and reflexivity . . . 99
5.5.2 Optimisations . . . 100
5.6 The decision procedure . . . 101
5.7 Experimental results . . . 103
5.8 Further work . . . 104

6 Conclusion **105**
6.1 Extend the model and tool . . . 105
6.2 Equivalence properties . . . 106
6.3 Compositionality . . . 106
6.4 Other attacker models . . . 107

Bibliography **109**

Chapter 1

Introduction

1.1 Some context

Until half a century ago cryptography was mostly used in a military or diplomatic context, to ensure secrecy of communications. At that time the use of cryptography was relatively simple, mainly encrypting messages was sufficient. Even then it oftentimes decided the fate of nations, from the fall of Mary Stuart [Sin99] to the successful decryption of the Enigma machine by Alan Turing that ended up being a keystone in the Allies' success during the Second World War [wik14]. With the rise of the Internet and the widespread usage of computers, the use of cryptography became much more complex and diverse. We started building complex protocols relying on the relatively simple building blocks that are cryptographic primitives like encryption, signature... Nowadays cryptographic protocols are used everywhere in our everyday life. Credit card payments on the Internet uses SSL/TLS for their security. Mobile phones use cryptographic protocols to ensure privacy and integrity of communications. With the adoption of electronic passports, we use cryptographic protocols to identify travellers at our borders. Even our political system relies on it with several countries using e-voting protocols for legally binding elections, like Estonia or even France. Ranging from the privacy of a vote or the secrecy of a phone conversation to our credit card number, we trust cryptographic protocols to protect a lot of very sensitive information. Moreover, when it comes to election or paying our taxes online, authenticity of the data transferred is also critical. In other terms, most of our privacy, and even some of the basic functions of our society rely on cryptographic protocols meeting their goal.

One valid question in this context is "is our trust in such protocols misguided". Sadly the answer to this question is not the one we could hope for. We could cite as example the Heartbleed vulnerability on SSL/TLS that may have exposed millions of user's passwords for various services, and let any evil-minded individual impersonate web services using this flawed implementation of SSL/TLS. Countless other examples exist, among others attacks on Google single sign-on[ACC+08]. This attack could have allowed a malicious service provider to gain access to personal information stored in any Google service. However, trust in cryptographic protocols is essential. Whole parts of the economic life rely heavily on such trust: cloud computing services offered by Google

or Amazon, countless online shops, banks... We need, at least, to be able as individuals to trust the result of our elections, to trust the fact that our taxes have been correctly filed. Moreover, with the increasing importance of cryptocurrencies, even the value of our money might one day depend on the faith we have in cryptographic protocols.

Considering the importance of security of cryptographic protocols, it is clear that designing a protocol and hoping that no one will be able to break it is not an option. We need formal guarantees on the correctness and safety of cryptographic protocols. We have been able to successfully build reliable software for several decades now, using both testing and verification approaches. We rely on critical software for driving some of our trains, for flying our planes or even our space shuttles, with very few critical failures since the Ariane crashes. Why is the picture so different for cryptographic protocols? For cryptographic protocols the testing approach is not an option. Indeed, in classical software, we can hope that if the software has passed sufficiently many tests, nothing bad will happen when actually using the software. However for cryptographic protocols if there is only one bad case, we can expect a clever attacker to find it. Another huge difference is that, for usual critical software, we are mostly interested in correctness of the software, while in cryptographic protocols we are dealing with much more complex properties. A protocol might be correct (i.e. produce the expected set of outputs given a certain set of inputs), but this does not mean that no information leaks from these outputs. Not only the outputs are important. Even if the messages exchanged in an e-voting protocol preserve the secrecy of messages, the control flow might leak the vote. For example the adversary may be able to force an error if the vote has a certain value. This gap is the reason for which we have to design specific techniques for proving cryptographic protocols.

1.2 State of the art

Until the 1990s cryptography was mostly a game of hide and seek between cryptologists and cryptanalysts. One would come up with a new protocol and give informal arguments for its security and cryptanalysts would try to break the protocol, often succeeding. Then a fix for the protocol would be proposed, followed by another try at breaking it and so on and so forth. Since then, tremendous progress has been made in the task of providing formal guarantees for both protocols and cryptographic primitives. The main step in being able to perform rigorous proofs is to understand what an attacker against a protocol may or may not do. Even for the basic capabilities such as intercepting and modifying messages, there has been some discussion. Should an attacker be able to do more than just eavesdrop communications, like injecting messages on the network? This first point has been settled in 1978 by Needham and Schroeder stating in [NS78]: "We assume that the intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material. While this may seem an extreme view, it is the only safe one when designing authentication protocols."

However, while the power of intercepting and forging messages is definitely a capability of any reasonable adversary, the computing power of the adversary remains unclear. Should it be an arbitrary Turing machine, should it be a

process only able to perform some particular operations? Indeed, reflecting this alternative, two main attacker models arose for proving protocols: the symbolic models and the computational models. Note that these are not the only two attacker models one can imagine; an adversary may have more capabilities than computational capabilities in trying to break protocols. For example, he might be able to observe other channels than just the messages sent around on the network, typically timing or power consumption [HMA+08, Koc96].

1.2.1 The symbolic models

Historically, the first proposal of a formal framework for proving protocols dates back to 1983 by Dolev and Yao in [DY83]. In this model the meaningful operations, which an attacker may perform, are applying the functions involved in the protocol. Typically encrypting, decrypting, projecting a pair... More precisely in the so called symbolic models the protocols and the attacker are processes in a concurrent process algebra, such as the applied π-calculus [AF01] or others [THG99, AG99].

In these models, the messages are modelled as abstract terms, living in a term algebra. Intuitively this means that a message records the information on how it was built. We are left with the task of specifying the attacker's capabilities. This is usually done by giving attacker rules for building messages. For example an attacker should be able to compute the pair of two messages he knows, or decrypt an encryption with a key he obtained:

$$\frac{u \quad v}{\langle u, v\rangle} \qquad\qquad \frac{k \quad \mathrm{enc}(m,k)}{m}$$

Note that a symbolic adversary is only able to perform operations that are specified by such rules (or any equivalent means). This means that, as long as we do not specify a capability of the adversary, the adversary does not have this capability. In other terms, one should be especially careful, when designing symbolic models, about the properties of the cryptographic primitives. Assume that given the encryptions of two plaintexts u, v, it is possible to compute the encryption of the pair $\langle u, v\rangle$. This is the case if we use some homomorphic encryptions. If we do not specify a rule

$$\frac{\mathrm{enc}(u,k) \quad \mathrm{enc}(v,k)}{\mathrm{enc}(\langle u, v\rangle, k)}$$

the symbolic adversary would not be able to use this homomorphic property. Intuitively, the cryptographic primitives are perfect in the sense that they only have the properties that are explicitly assumed.

In these models, the security properties can be divided between reachability properties and equivalence properties. The former states that in a certain state the adversary should not be able to derive a secret s, or if a certain point in the protocol is reached, the principals should agree on some value, or similar properties. This is typically a good way to define secrecy and agreement properties. The later is defined by stating that no adversary should be able to distinguish two processes. Anonymity is a typical example of equivalence properties. For example, voter anonymity[KR05] where the adversary should not be able to notice a swapping of the votes in a voting process.

The main advantage of these symbolic models is that they can be automated. There are lots of tools that check trace properties like ProVerif [Bla01], Scyther [Cre08], or Avispa[ABB+05]. There has been a lot of work towards automated verification of equivalence properties as well, leading to tools like Apte [CCD10]. The main reason of the success of the automation of this model is that the adversarial capabilities are specified in a relatively simple logic, and the messages live in a simple term algebra. Therefore, it is quite close to widely studied problems in automated reasoning.

1.2.2 The computational model

In the computational model the attacker is not an abstract process dealing with unambiguous terms but a probabilistic polynomial time Turing machine (PPT) that deals with bitstrings. This model dates back to [GM84]. The key idea is to define security as a game that the adversary should not be able to win. This model allowed to define the security of cryptographic primitives in a rigorous way. This leads to various levels of security that can be expected from cryptographic primitives against attackers that are more or less powerful. Typical examples are IND-CPA security ([GM84]) or IND-CCA security ([RS91]). The former states that an encryption scheme should not leak information on the plaintext in the presence of an adversary that is able to obtain ciphertexts. The latter models the fact that an adversary should not be able to obtain information from a ciphertext even if he has access to a decryption oracle. Note that these rigorous definitions of security for the primitives is a key difference with the symbolic models. In the symbolic model the cryptographic primitives are assumed perfect while in the computational model the cryptographic primitives are assumed to satisfy no more than what is explicitly stated by the security property. We will not however consider the proofs of cryptographic primitives in this thesis. We assume that the cryptographic primitives do have some security properties and use them to prove protocols.

Let us go a bit more into the details of the computational model. As an example, say that the protocol P should ensure the secrecy of some value s. Usually, the computational secrecy is defined as the indistinguishability of s from a random value. We may also consider a weaker notion of secrecy, defined as follows. Given a PPT $\mathcal{A}$, we say that $\mathcal{A}$ wins the secrecy game if $\mathcal{A}^P$ ($\mathcal{A}$ interacting with the protocol P as an oracle) is able to compute s. We say that the protocol is secure if no PPT may win this game with non-negligible probability. The assumptions on the cryptographic primitives are also defined by games. For example in the IND-CPA game the adversary is given access to either a real encryption oracle or a fake encryption oracle that encrypts zeros instead of encrypting its input. The adversary wins this game if he is able to distinguish between the real and the fake oracle. A proof of P typically reduces winning the IND-CPA game to winning the secrecy game against the protocol. In other words, a proof consists in providing a PPT $\mathcal{B}$ simulator such that, given an adversary $\mathcal{A}$ that breaks the secrecy game, then $\mathcal{B}^A$ breaks the IND-CPA game.

The main advantage of this model is that the attacker is much closer to what a "real life" attacker would be. If needed to prove a protocol, cryptographic properties must be explicitly assumed. Indeed, as the proof works by reduction of a hard problem, if there is no hard problem, there may be no reduction. In

other words, the modelling of the attacker is not an issue. However finding the minimal hardness assumptions can be difficult.

1.3 Comparison and critic of existing models

1.3.1 Symbolic models vs. computational model

First of all, let us consider the accuracy of the computational and symbolic models. The first difference between these models is the way messages are represented. The symbolic models abstract messages as terms, therefore it may overlook some confusion between messages. For example, if we only assume that an encryption scheme is IND-CPA secure there is no reason for the set of encryptions with some key to be disjoint from the set of encryptions with another key. However, any reasonable term algebra would assume that. This means that, in the symbolic models, we may miss some attacks that take advantage of such confusions. Conversely, in the computational model, the messages are bitstrings, which is precisely what the messages are in "real life", therefore we may not miss this kind of attacks. A related problem is the capabilities of the adversary. As mentioned earlier, a symbolic adversary may only compute messages according to a fixed set of rules. This leads to the problem mentioned earlier: if an encryption scheme is malleable, this should be reflected in the capabilities of the adversary. However, if an encryption scheme is only assumed to be IND-CPA, it may be malleable in very different ways. It appears to be a very complicated task to model them all. In other words, it seems close to impossible to deal with under-specified cryptographic primitives in the symbolic model, while the properties of cryptographic primitives are almost never fully described and the security definitions leave numerous degrees of freedom. This problem does not arise in the computational model. Indeed, the game based proofs only rely on the (proven or assumed) properties of the cryptographic primitives.

As a short example of this, let us examine the Needham-Schroeder-Lowe (NSL) protocol[Low95]. Informally, the NSL protocol between agents A and B is meant to ensure that if both agents complete the protocol successfully, they should agree on the values of some secrets. It is described as follows:

$$\begin{array}{rcl} A & \rightarrow & B: \quad \{n_A, A\}_{pk_B} \\ B & \rightarrow & A: \quad \{n_A, n_B, B\}_{pk_A} \\ A & \rightarrow & B: \quad \{n_B\}_{pk_B} \end{array}$$

where $\{x\}_{pk_Y}$ denotes the encryption of x with the public key of agent Y. If one agent completes the protocol successfully, A and B should agree on the random values n_A and n_B and these values should be secret. Now, if the encryption scheme is malleable enough – which is the case using an El-Gamal[Gam85] encryption with a particular implementation of pairing – an adversary can obtain $\{n_A, I\}_{pk_B}$ from $\{n_A, A\}_{pk_B}$, leading to the following man in the middle attack[War03]:

$$\begin{array}{rcl} A & \rightarrow & I(B): \quad \{n_A, A\}_{pk_B} \\ I & \rightarrow & B: \quad \{n_A, I\}_{pk_B} \\ B & \rightarrow & I: \quad \{n_A, n_B, B\}_{pk_I} \\ I(B) & \rightarrow & A: \quad \{n_A, n_B, B\}_{pk_B} \\ A & \rightarrow & B: \quad \{n_B\}_{pk_B} \end{array}$$

From the point of view of A the protocol was completed correctly, however the adversary I has learned the value of n_A and n_B. This protocol has been proven secure in the symbolic model several times. However, all these proofs implicitly assume that the encryption scheme is not malleable. Or, more accurately, they assume adversarial capabilities that do not take into account the possible malleability of the encryption scheme. On the other hand, trying to prove this protocol secure in the computational model without assuming any form of non-malleability of the encryption scheme is not possible. This example is a good illustration of the gap between the computational and the symbolic models. While the computational model has precise cryptographic assumptions, in the symbolic models the adversarial capabilities can not take into account all possibilities that are open to a real life adversary.

Let us now compare the models from the viewpoint of the proof search. On the one hand, in the symbolic models, as mentioned earlier, the proofs are often completely automated. It is also worth noting that usually proofs in the symbolic models are done by applying the attacker's rules up to saturation. Therefore disproving the security property often yields a sequence of attacker rules applications leading to an attack. Indeed the symbolic models have been quite successful at finding attacks since their very beginning. As an example let us mention Lowe's attack[Low95] on the Needham-Schroeder protocol[NS78]. The Needham Schroeder protocol was designed in 1978 and was believed secure until 1995 when G. Lowe, while trying to prove the protocol secure in the symbolic model, found a man in the middle attack. Moreover, the proofs carried out in the symbolic models are mainly reasonably simple proofs using a simple logic. This fact, together with the automation of the process gives a very good level of confidence in such proofs, although it is still difficult to obtain independently verifiable proof scripts.

On the other hand, in the computational model, while there are proof assistants such as EasyCrypt[BGHB11] or CryptoVerif[Bla06], there is no fully automated prover so far and little hope of designing one in the near future. Not only the proofs require some user interaction, but a failure in the proof search is difficult to interpret. Indeed when trying to prove a protocol in the computational model, the goal is to find the correct reduction. Therefore if one fails to prove a protocol in the computational model this failure does not necessarily yield an attack. It might be that the proof is simply too difficult to find.

Moreover, most proofs in the computational model are still performed by hand and not machine checked. It is hard to have complete trust in such proofs. Indeed proofs in the computational model tend to be quite involved, as it is necessary to take into account probabilities and complexity of the reductions. There are several examples of computational proofs of protocols or primitives which ended-up being wrong. One of the most striking example might be the case of the RSA-OAEP[BR94] security proof. The RSA-OAEP scheme was originally proven secure by Bellare and Rogaway in [BR94]. The proof was flawed and was successively corrected by Shoup[Sho02], Fujisaki, Okamoto, Pointcheval and Stern[FOPS04] and then Pointcheval[Poi05]. In the end a correct, machine checked proof was finally established by Barthe, Grégoire, Lakhnech and Zanella Béguelin in [BGLB11]. The main reason for such mistakes is the fact that proofs of cryptographic protocols in the computational model do not usually include full details because of their complexity.

Let us sum up the previous part. The advantage of proofs in the symbolic

models is that we have automation and attack reconstruction. On the other hand the computational model is more accurate, in the sense that it gives precise assumptions on the cryptographic primitives. Therefore a proof of protocols in the computational model yields stronger guarantees.

1.3.2 Computational soundness

The aim of computational soundness is to get the best of the two worlds. This area of research was started in the year 2000 by Abadi and Rogaway[AR02]. The main idea is to prove that the symbolic attacker is at least as powerful as the computational one. Then any symbolic security proof yields the computational security. (Though, the absence of a symbolic security proof does not necessarily yield a computational attack). This area of research has been developed quite successfully since then. The original result by Abadi and Rogaway was obtained for a passive adversary – an adversary that may only observe the run of the protocol. The subsequent results were obtained for active adversaries. Let us cite as examples among many others [BP04, MW04, CC08]. These results have been obtained for various security goals, mainly trace properties and equivalence, and various sets of primitives.

All these soundness results come to the cost of strong implementation hypotheses, and restrictions on the class of protocols:

- Strong cryptographic hypotheses. In order to obtain a trace mapping result, strong assumptions on the behaviour of the cryptographic primitives are needed, such as IND-CCA or strong integrity.
- Parsing assumptions: every function should append an unambiguous tag to its result. The reason for this assumption is to avoid confusion between messages. The only result dropping the parsing assumption is [CHKS12], which comes at the cost of a very complex proof.
- Absence of key-cycles: the protocol may never produce the encryption of an honest key with itself or similar patterns. This can be dropped at the cost of stronger cryptographic assumptions, for example the KDM assumption[BRS02]. However practical encryption schemes do not satisfy this assumption.
- Absence of dishonest key generation: all keys are randomly generated using the key generation algorithm. The assumption implies that the adversary may not generate its own keys. It would typically be enforced using a key distribution authority. In practice, as far as symmetric keys are considered, keys are generated by the agents without any further check. This contradicts the honest keys hypothesis. We showed in a previous work[CCS12] (which is not presented here) that this hypothesis may be dropped. However the cost of dropping this hypothesis is considering a much more complex symbolic model. It seems hard to automate proofs in such a model. Another way to drop this hypothesis is to strengthen considerably the cryptographic assumption as done in [BMU12]. The cryptographic assumption needed for such a strong result are far from being satisfied by practical cryptographic primitives.
- The commitment problem: there are several impossibility results[BP05] for the computational soundness of primitives such as exclusive or, hash functions,...

It is unclear whether any actual implementation satisfies all the hypotheses of

a soundness result. However, these results give a good intuition of what are the necessary assumptions for a symbolic proof to yield computational guarantees.

Let us have a closer look at proofs of computational soundness. The goal is to obtain a trace mapping lemma. This is done by showing that if a trace exists in the computational model but not in the symbolic model, the cryptographic hypotheses are broken. The main difficulty in this case is the fact that the reduction must be uniform. In other terms there should be one reduction that works for all protocols. There are two consequences of this remark. First, computational soundness proofs are very heavy, even by the standard of proofs in the computational model. Second, the proofs are very strongly connected to the process algebra we consider. This means that computational soundness proofs are not modular at all. If one has proven soundness for a process algebra allowing symmetric encryption and signature, then adding asymmetric encryption means doing a completely new soundness proof. Considering that the number of cryptographic primitives increases with time, this means doing heavier and heavier proofs over time. This modularity problem has been addressed by Böhl, Cortier and Warinschi in [BCW13]. This paper describes conditions on primitives for having a composable soundness theorem, however the implementation hypotheses are once again quite strong.

The main reason of such complexity in the computational soundness proofs can be summed up as follows. In the computational model, the adversarial capabilities are defined by what the adversary can not do. On the other hand, in the symbolic models, the adversarial capabilities are defined by what the adversary can do. In other terms in the computational model the adversarial capabilities are defined as a greatest fixed point while in the symbolic models they are defined as a least fixed point. The difficulties in soundness results is to make sure that these two fixed points actually coincide.

1.3.3 The computationally complete symbolic attacker

Are we stuck with this unsatisfactory situation? Is it impossible to combine the computational guarantees with the nice properties of the symbolic models without having unrealistic cryptographic hypotheses? The answer given by Bana and Comon in [BC12] is no. The key idea in this paper is defining a new symbolic attacker – namely the computationally complete symbolic attacker – that considers the attacker as a greatest fixed point.

In this model, we specify what the attacker cannot do by axioms in first order logic. These axioms reflect quite closely the cryptographic hypotheses. Then a transition of the protocol is possible if the condition associated with this transition is consistent with the axioms. We define security in this model as the unsatisfiability of the negation of the security property together with the trace conditions and the axioms. From this definition, the satisfiability means that there is a model, which is an attack witness. It follows that this model is very convenient for finding attacks as demonstrated in [BAS12, BHO13]. Moreover, checking security amounts to checking (un)satisfiability of sets of first order formulae. This problem has been widely studied in the last 50 years, and as demonstrated in chapter 4 we can reuse proof techniques of this field to prove decidability of the particular subset of first order logic we study.

Having defined this model – and stated that it has the desirable properties of a symbolic model – we are left with precising its relationship with the compu-

tational model. This relationship mainly amounts to defining a computational semantics of first order logic. This part is done by seeing the computational model as a certain form of Kripke model and using an embedding of first order logic in the S4 modal logic due to Fitting[Fit70]. If the first order axioms are proved to be valid in such a computational semantics (thanks to the cryptographic hypotheses), then the symbolic proof yields computational security. The validity of the axioms with respect to the computational semantics are relatively simple cryptographic proofs, as will be demonstrated in chapter 3. It follows that modularity is for free in this model: if an axiom A is sound with respect to the cryptographic properties of f, and axiom B is sound with respect to the cryptographic properties of g, the conjunction of these axioms is sound with respect to the joint cryptographic properties. Therefore, unlike usual soundness results, we do not have to redo the whole proof if we want to add a primitive.

This model also avoids parsing assumptions, that are customary even in computational proofs. This way we may find new attacks that rely on confusions between messages. We could cite as example the attack from [BC12] on the NSL protocol. This attack relies on the fact that a random number might be confused with a pair. This attack was not found before because all proofs implicitly assumed that such a confusion was impossible. However it is not hard to think of implementations of the pair that would allow such an attack.

The main drawback of this model with respect to the computational model is that it does not provide explicit bounds on the reduction. In the computational model, when a proof is completed, a close look at the reduction provides a polynomial p such that if it is impossible to break the security of the cryptographic primitive in time t, then it is impossible to break the protocol in time $p(t)$. When proving protocols secure against the computationally complete symbolic attacker we only know that such a polynomial exists, but we have no way (so far) of actually computing it. Note that this is also a drawback for classical soundness results. We do not think that this loss of precision on the bound can be avoided without having a much more complex symbolic model, which involves probabilities.

1.3.4 Other symbolic models for a computational attacker

There are, to the best of our knowledge, only two other symbolic models that give sound axioms that are used for the proofs. The first one is the Protocol Composition Logic (PCL)[DDM$^+$05] and the second one is F7[FKS11]. They are both computationally sound with much weaker cryptographic hypotheses than usual soundness results. The way they define security is quite similar, PCL uses Hoare logic, while F7 uses refinement types, which is intuitively the combination of types and Hoare logic. They are both successful at proving protocols, even large protocols like TLS[BFCZ12] or Kerberos[RDM07]. One key difference with the Computationally Complete Symbolic Attacker is that security is not proven by inconsistency. In their approach the security goal should follow from the axioms and the Hoare like inference rules. This means that if one fails to prove a protocol in these models, he does not necessarily obtain an attack. He only knows that his axioms were not strong enough (or that he failed to find a proof). It should also be noted that there is no automation of these models: it is the user who writes type annotations in F7 and Hoare triples in PCL. The logic used in these models is much more complex than the one used in our approach,

where we use only a simple fragment of first order logic. Therefore it is much more difficult to automate.

1.4 Our contribution

The original model from Bana and Comon[BC12] defines a computational semantics of first order logic from scratch. In order to perform proofs it is therefore needed to prove that this computational semantics is equivalent to the first order semantics of the logic. Moreover, due the complexity of this computational semantics the soundness proofs of the axioms are hard to check. A more elegant formulation is introduced in [BHO13]. We follow this formulation and detail the model in 2. With this definition, there is no need to perform complex proofs on the computational semantics as it is simply a Kripke semantics. Moreover, this simplification of the model yields simpler soundness proofs for the axioms.

We provide here with a formal statement of axioms for symmetric and asymmetric encryption in chapter 3. The axioms we state and prove sound are more general than the axioms from [BC12]. In more details, our contribution here is threefold. For the "core" axioms, that are independent of any cryptographic hypotheses we (apart from a generalisation) give new soundness proofs of the original axioms from Bana and Comon. For the axiom representing the secrecy property of the encryption, we also consider the case of symmetric encryption, which is not encompassed by the results in [BC12]. This extension is non-trivial, as it requires an unexpected additional hypothesis. The additional hypothesis is the "which-key concealing property", which states that an encryption should not disclose information on the encryption key. The last class of axioms we propose are integrity axioms, both for non-malleability and unforgeability of encryptions. These two axioms are different from the ones proposed in [BAS12] and [BHO13]. We chose another approach for tackling the problems that is both more convenient for performing soundness proofs, and for automating the model. Moreover, we believe that the proof techniques developed in chapter 3 could be reused to prove axioms for other primitives like signatures or hash functions.

One question left open in [BC12] is the automatability of this symbolic model. Even if the problem remains a satisfiability check for sets of first order formulae, it has several unusual specificities. There was hope that such a fragment would be decidable: if the set of axioms is saturated, then the satisfiability could even be tractable[BG01]. There is however a difficulty, since we use a variadic predicate symbol (the "computability" predicate) as well as ground equations and constrained formulae. Nevertheless, having an *efficient* decision procedure for this problem is actually crucial as proving security in this model means performing one satisfiability check per trace of the protocol. We give such a procedure in chapter 4.

The last part of this thesis describes our tool (Scary) implementing (a variant of) this decision procedure, together with experimental results. We only experimented this tool on a few protocols so far, but it has already yielded a new attack on Andrew's Secure RPC[Sat89], and we hope to find many more attacks in the future.

Chapter 2

Formal setting

In this chapter we define the formal setting in which we prove protocols. Most symbolic framework (for example the applied π-calculus[AF01] or strand spaces[THG99]) specify what an adversary is able to do. Typically, a symbolic adversary has finitely many rules to build messages from its knowledge. On the other hand, when considering computational cryptography, instead of listing the moves that an adversary may perform (it is an arbitrary PPT), we restrict its capabilities by stating that it should not be able to break the cryptographic properties of the primitives. In other words, classical symbolic adversaries have finitely many allowed moves while computational adversaries have finitely many forbidden moves.

The model we present here aims at defining a symbolic adversary (or more accurately a symbolic notion of security) that mimics the computational model in the sense that it specifies what an adversary can not do. The key idea of this model is to allow any adversarial move, as long as it does not contradict some axioms reflecting the cryptographic properties. It is however a symbolic model in the sense that it abstracts the probabilities and Turing machines. The fact that proving security mainly amounts to checking consistency of sets of first order formulae means that in the task of proving a protocol secure, we will be able to reuse some existing ideas of automated reasoning, as will be detailed in chapter 4.

This setting was originally defined by Gergeï Bana and Hubert Comon-Lundh in [BC12]. In a following paper [BHO13] the authors introduce a more elegant formulation in terms of Kripke semantics. In this chapter we rely on this presentation. In order to prove security in this framework, we need to prove that, for each trace, a set of ground formulae corresponding to this trace together with the negation of the security property and some axioms is inconsistent. This security proof ensures computational security if the axioms are sound with respect to the computational interpretation of the predicates and functions.

Intuitively the ground formulae encode the conditions under which a trace is executable (basically gathering all tests done by the protocol together with conditions stating that adversarial messages should be computable from the relevant knowledge). The axioms encode the properties of the predicates and functions (typically the cryptographic properties). If these formulae and axioms can be satisfied together with the negation of the security property, it then means that the trace is executable by an adversary and then the security prop-

erty is broken. There might be two reasons for this fact: either the axioms do not account for all the properties of the adversary, allowing for a false attack, in which case the axioms should be strengthened; or the protocol is flawed. Conversely, if no attack is found on a trace it means that the assumptions on the primitives ensure that the trace is either non executable or the security property holds on this trace.

We start by defining the first order semantics of the logic we consider. This logic will be used to perform security proofs in an abstract way. It is also the logic that is used to guard transitions of the protocols. As we want to prove that security proofs in the model entail computational guarantees, we proceed to a computational semantics of the logic. In order to do this we need to introduce probabilities in the semantics. This is formulated in terms of Kripke structures, where worlds are non-negligible sets of random samplings. The intuition governing this construction is that we want to consider all possible sets of samplings that are large enough to provide with a computational attack. The relationship between the computational logic and the first order logic is obtained thanks to an embedding of first order logic into a modal logic (namely S_4), due to Fitting[Fit70]. Having defined properly both a computational and a first order semantics, we move on to defining the execution model of protocols and the soundness result stating that if there is no symbolic attack on a protocol, then it is computationally secure (provided that the axioms used to prove this protocol are computationally sound).

2.1 First order semantics

Let $\mathcal{F}$ be a finite set of function symbols (together with their arity), $\mathcal{C}$ a sorted set of constants and $\mathcal{P}$ be a finite set of predicate symbols together with their arity. We write f/a for the function or predicate f with arity a. $T(\mathcal{F}, \mathcal{C})$ is the set of ground terms built on $\mathcal{F} \cup \mathcal{C}$ (which is assumed to contain at least one constant in order to actually be able to build terms).

Example 2.1. We take as our running example the term algebra of the tool presented in chapter 5, built on:

- the set of functions:

  $$\mathcal{F} = \{\text{senc}/3, \text{sdec}/2, \text{aenc}/3, \text{adec}/2, \text{pk}/1, \text{pair}/2, \pi_1/1, \pi_2/1\}$$

  composed of symmetric and asymmetric encryption and decryption, asymmetric public key derivation, pairing and projections. We also consider countably many functions $\{h_i/i | i \in \mathbb{N}\}$ we call handles. These functions stand for the adversarial messages.
- the set of constants $\mathcal{C}$ with five sorts $\mathsf{nonce}, \mathsf{name}, \mathsf{skey}, \mathsf{akey}$ standing for random nonces, names, symmetric keys and asymmetric private keys, each having countably many members.

For non-ground terms we need to consider variables, we consider a set of variables $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_F$ where $\mathcal{X}_C$ is a set of sorted variables (containing countably many variables for each sort of $\mathcal{C}$) and $\mathcal{X}_F$ a disjoint set of variables that will stand for terms without restrictions of sort. We define $T(\mathcal{F}, \mathcal{C}, \mathcal{X})$ as the set of terms built on $\mathcal{F}$ and a set of variable symbols $\mathcal{X}$.

We define as usual the function symbol at position p in term t as t_p and the subterm rooted at position p as $t_{|_p}$. We denote by $\text{st}(t)$ the set of subterms of t.

We also use set variables (written using upper case letters $X, Y, Z, ...$) ranging in a set $\mathcal{SX}$ and a function symbol, denoted by a semicolon, for set union. *Extended terms* $ET(\mathcal{F}, \mathcal{C}, \mathcal{X}, \mathcal{SX})$ are expressions $s_1; \ldots; s_n$ where $s_i \in T(\mathcal{F}, \mathcal{X}) \cup \mathcal{SX}$. As a shortcut, when $n = 0$ in the previous definition we denote the extended term as $\emptyset$. A *basic ordering* is an ordering on terms, which is:

1. Compatible with substitutions and
2. such that, for every ground term t, the number of terms smaller than t is polynomial in the size of t. (An example of such an ordering is the subterm ordering).

Atomic formulae are of the following forms:

- $P(t_1, \ldots, t_n)$ where $P \in \mathcal{P}$ and $t_1, \ldots, t_n \in T(\mathcal{F}, \mathcal{C}, \mathcal{X})$
- $t_1 = t_2$ where $t_1, t_2 \in T(\mathcal{F}, \mathcal{X})$
- $S \triangleright t$ where $t \in T(\mathcal{F}, \mathcal{X})$ and $S \in ET(\mathcal{F}, \mathcal{C}, \mathcal{X}, \mathcal{SX})$.

The unusual $\triangleright$ predicate intuitively means that knowing the set of terms S, it is possible for an adversary to build term t. We do have to consider sets of terms on the left hand side of this predicate as restricting this predicate to be a n-ary predicate would mean bounding the number of messages sent by a protocol.

We consider clauses that are built on these atomic formulae. The axioms for the set theory ACIN (associativity, commutativity, idempotence and neutral element $\emptyset$) are implicitly assumed without mention on the left side of the $\triangleright$. Intuitively, this is because, the set S represents the knowledge of an adversary, and, as such, order or duplication of the terms is not relevant. We also implicitly assume, if not stated otherwise, the axioms for equality (equality is a congruence) in what follows.

Given an extended term S and a substitution σ, mapping variables of $\mathcal{SX}$ to finite subsets of $T(\mathcal{F}, \mathcal{C})$, variables in $\mathcal{X}_C$ to constants with the corresponding sorts and variables of $\mathcal{X}$ to terms in $T(\mathcal{F}, \mathcal{C})$, $S\sigma$ is defined by $\emptyset\sigma = \emptyset$, $(s; S)\sigma = \{s\sigma\} \cup S\sigma$ if $s \in T(\mathcal{F}, \mathcal{C}, \mathcal{X})$, and $(X; S)\sigma = X\sigma \cup S\sigma$ if $X \in \mathcal{SX}$.

2.1.1 Constraints

A *constraint* Γ is a formula interpreted as a subset of $((T(\mathcal{F}))^*)^n$ (n-uples of finite sets of ground terms) if n is the number of free variables of Γ. We write $S_1, \ldots, S_n \models \Gamma$ when $(S_1, \ldots, S_n)$ belongs to this interpretation. A *constrained clause* is a pair of a clause and a constraint, which is written $\phi \quad \| \quad \Gamma$. Given a constrained clause $\phi \quad \| \quad \Gamma$, we let $[\![\phi \| \Gamma]\!] = \{\phi\sigma \| \sigma \text{ satisfies } \Gamma\}$. A model of $\phi \quad \| \quad \Gamma$ is, by definition, a model of $[\![\phi \| \Gamma]\!]$.

These constraints will typically be used in axioms to ensure that axioms are only applied to frames where the key usage complies to what is authorised by the cryptographic properties, or to check whether some value is not guessable from a set of terms.

Example 2.2. Let us give a first example of constraint, the freshness constraint written $\text{fresh}(n, X)$ where n is a constant. The semantics of this constraint is

$$S \models \text{fresh}(n, X) \text{ iff } \forall t \in S.n \notin \text{st}(t)$$

This constraint is used in the freshness axiom that states that a random value should not be computable from a set of terms that does not depend on it (for more details see chapter 3)

$$\neg X \triangleright n \parallel \text{fresh}(n, X)$$

which represents the set of clauses $\neg S \triangleright n$ where n does not appear in S.

Definition 2.3. A constraint Γ is *monotone* if

- if $S_1, \ldots, S_n \models \Gamma$ and, for every i, $S'_i \subseteq S_i$, then $S'_1, \ldots, S'_n \models \Gamma$
- if $S_1, \ldots, S_n \models \Gamma$ and $S'_1, \ldots, S'_n \models \Gamma$, then $S_1 \cup S'_1, \ldots, S_n \cup S'_n \models \Gamma$.

Note that the conjunction of two monotone constraints is a monotone constraint, it is not, however, true of the disjunction of two monotone constraints.

Let us remark that a typical monotone constraint is a property of terms and not a property of sets of terms. For example, the aforementioned freshness constraint is monotone, as it is a property of the individual terms in S rather than a property of S as a set. Note that all flavors of plaintext freshness constraints (stating that a key should not appear in any terms of S except for some positions) are monotone.

Monotonicity of constraints are be used in chapter 4 to ensure that the strategy is correct. Note however that the strongest constraints we consider in our tool and in chapter 3 are not monotone, however the simplest constraints which are often enough to prove protocols that make use of only asymmetric encryption or pre-shared keys do verify this property.

Let us give a short example of a non-monotone constraint.

Example 2.4. The key usability constraint, properly defined in chapter 3, is not a monotone constraint. Intuitively this constraint is inductively defined by the fact that a key k is usable in a set of terms S if it only appears encrypted by keys that are themselves usable. Typically k is usable in $S_1 = \{k\}^r_{k'}$ as it is safely encrypted by key k' that is not disclosed in S_1, it is also usable in $S_2 = k'$ as it does not appear in S_2. However k is not usable in $S_1 \cup S_2 = \{k\}^r_{k'}; k'$, as k' is not usable itself. Therefore this key usability is not monotone.

2.1.2 Semantics

A first-order structure $\mathcal{S}$ is defined as usual. We only require that $=$ is interpreted as a congruence relation on the interpretation domain $D_{\mathcal{S}}$ and that $\triangleright$ is interpreted as a relation between the finite subsets of $D_{\mathcal{S}}$ and $D_{\mathcal{S}}$.

The only somewhat unusual part is the satisfaction for constrained clauses, which is defined as the satisfaction of all instances of the clause satisfying the constraint:

$$\mathcal{S} \models \phi \| \Gamma \text{ iff } \mathcal{S} \models [\![\phi \| \Gamma]\!]$$

Let us give a short example of this semantics.

Example 2.5. Let us consider the following first order structure $\mathcal{S}$. Its interpretation domain $\mathcal{D}_{\mathcal{S}} = T(\mathcal{F}, \mathcal{C})$ with $=$ interpreted as the trivial congruence. We fix n. Let us now define the $\triangleright$ relation as: $S \triangleright n$ if and only if $\neg\, \mathrm{fresh}(n, S)$. We now have

$$\mathcal{S} \models \neg X \triangleright n \| \,\mathrm{fresh}(n, X)$$

However, if we change the interpretation of $\triangleright$, adding $\emptyset \triangleright n$, $\mathcal{S}$ is not a model of $\neg X \triangleright n \| \,\mathrm{fresh}(n, X)$ anymore. Indeed we have $\mathcal{S} \not\models \neg\emptyset \triangleright n$ and the clause $\neg\emptyset \triangleright n$ belongs to $[\![\neg X \triangleright n \| \,\mathrm{fresh}(n, X)]\!]$

2.2 Kripke semantics

In order to give a computational meaning to this framework, we need to define a computational semantics. Following [BHO13] we define here a Kripke model that captures the capabilities of a computational adversary. The worlds are sets of random samplings, representing adversarial and protocol's randomness. The successor relation is simply the inclusion relation between these sets of random samplings. Using an embedding due to Fitting[Fit70] we have the result that if a set of formulae holds in the Kripke model considered here, then it has a first order model. Note that this embedding, together with the fact that a computational attack on a trace ensures that the corresponding set of formulae has a Kripke model, yields the computational soundness result we wanted.

2.2.1 Samplings, functions and constants

We start by defining the worlds of the Kripke model. We consider first order structures with domain $\mathrm{map}(S, \{0,1\}^*)$, the functions from a non-negligible subset S of all the possible random samplings to the bitstrings. In this thesis, we assume that these functions are implemented by probabilistic polynomial time Turing machine (PPT) seen as mapping from their random tape to their output. Some of the functions, constants and predicates have a fixed interpretation while others do not. Let us be more precise and start by defining what a sampling is.

Non-negligible sampling families

A random sampling is a family of countably many infinite sequences of bits indexed by the union of $\mathbb{N}$ and $\mathcal{C}$. Intuitively the infinite sequence of bits labelled by constants is used to compute the interpretation of constants, and the remaining sequences are used as random tapes for the machines that are witness to the satisfaction of the $\triangleright$ predicate.

Definition 2.6. A *random sampling* $(r^i)_{i\in\mathcal{C}\cup\mathbb{N}}$ is an element of the sampling space

$$\mathrm{Sspace} = (\{0,1\}^{\mathbb{N}})^{\mathcal{C}\cup\mathbb{N}}$$

We say that r^i is the *random tape* i. For $l \in \mathbb{N}$ we write r^i_l the initial segment of length l of r^i.

Now that the sampling space is defined, we need to specify what kind of subsets of the sampling space we want to consider. Note that a set of formulae contains only finitely many constants and finitely many $\triangleright$ statements therefore we only need finitely many random tapes. Moreover, all the machines used to compute constants or be witnesses for a $\triangleright$ statement terminate and therefore only use finitely many random tape bits. More formally we define uniform subsets as follows.

Definition 2.7. A sampling set S is (k,l)-*uniform* if the two following conditions are satisfied:

1. it can be written as $S_K \times \left(\{0,1\}^{\mathbb{N}}\right)^{(\mathcal{C}\cup\mathbb{N})\setminus K}$ with $\#K = k$. This condition states that only the tapes in K are restricted.

2. if $(r^1, \dots, r^k) \in S_K$, we have $(r^1_l.s^1, \dots, r^k_l.s^k) \in S_K$ for all $s^1, \dots, s^k \in \{0,1\}^{\mathbb{N}}$. This condition states that only the l first bits of any tape are restricted.

We say that a sampling set is *uniform* if it is (k,l)-uniform for some integers k, l. We denote by $\mathcal{U}$ the collection of uniform sampling sets.

As we want to define a measure on (uniform) sampling sets, we show that these form an algebra.

Proposition 2.8. *The collection $\mathcal{U}$ is stable by union, intersection and complement and contains* Sspace *and $\emptyset$.*

Proof. Let us start by remarking that Sspace and $\emptyset$ are $(0,0)$ uniform sampling sets.

Let S_1 be a (k_1, l_1) uniform set, and S_2 be a (k_2, l_2) uniform set. Let us write $S_1 = S'_1 \times (\{0,1\}^{\mathbb{N}})^{(\mathcal{C}\cup\mathbb{N})\setminus K_1}$ and $S_2 = S'_2 \times (\{0,1\}^{\mathbb{N}})^{(\mathcal{C}\cup\mathbb{N})\setminus K_2}$. Let $K = K_1 \cup K_2$, $k = \#K$ and $l = \max(l_1, l_2)$.

We show that $S_1 \cup S_2$ and $S_1 \cap S_2$ are (k,l) uniform. Note that we may write $S_1 = S''_1 \times (\{0,1\}^{\mathbb{N}})^{(\mathcal{C}\cup\mathbb{N})\setminus K}$ where $S''_1 = S'_1 \times (\{0,1\}^{\mathbb{N}})^{K\setminus K_1}$ abides by condition 2. We have $S_1 \cup S_2 = (S''_1 \cup S''_2) \times (\{0,1\}^{\mathbb{N}})^{(\mathcal{C}\cup\mathbb{N})\setminus K}$, it is now enough to note that condition 2 holds on $(S''_1 \cup S''_2)$. Similarly $S_1 \cap S_2 = (S''_1 \cap S''_2) \times (\{0,1\}^{\mathbb{N}})^{(\mathcal{C}\cup\mathbb{N})\setminus K}$ and condition 2 holds on $(S''_1 \cap S''_2)$.

The case of complement is even simpler: $S^c_1 = S'^c_1 \times (\{0,1\}^{\mathbb{N}})^{(\mathcal{C}\cup\mathbb{N})\setminus K_1}$ and condition 2 trivially holds on S'^c_1. □

Note that cryptographic properties provide asymptotic conditions on the attacker's success as the security parameter grows. Therefore we can not simply work with sampling sets. We need one sampling per security parameter, which we call a sampling family. Moreover, we also have to reflect the fact that the attacker's success probability may be negligible: we need to define sampling families that do not become arbitrarily small.

Let us start by defining the size of a sampling set. It roughly corresponds to the proportion of all samplings that are covered by this sampling set. We will define a measure μ on the sampling space that ensures that all uniform samplings are measurable, and that the image measure's by the projections on a finite set of tapes, truncated to a finite number of bits is the uniform measure.

Let us start by properly defining the projections:

$$\begin{array}{rccc} p_{K,l} \ : & \text{Sspace} & \to & (\{0,1\}^l)^K \\ & \left((r^i)_{i\in\mathcal{C}\cup\mathbb{N}}\right) & \mapsto & (r^i_l)_{i\in K} \end{array}$$

Let S be a (k,l) uniform sampling set, let us write $S = S_K \times \left(\{0,1\}^{\mathbb{N}}\right)^{(\mathcal{C}\cup\mathbb{N})\setminus K}$ with $\#K = k$. We define

$$\mu(S) = 2^{-kl} \# p_{K,l}(S)$$

Remark 2.9. The previous definition is independent of the choice of K and l. It is enough to notice that if S is (k,l) uniform and $S = S_K \times \left(\{0,1\}^{\mathbb{N}}\right)^{(\mathcal{C}\cup\mathbb{N})\setminus K}$ with $\#K = k$ we can increase K and l without changing the definition of μ. Let us consider K' such that $K' \cap K = \emptyset$ and $\#K' = k'$. Now, we get

$$p_{K\cup K', l+r}(S) = \left\{ (r^i_l.s_i)_{i\in K\cup K'} | s_i \in \{0,1\}^r \text{ and } (r^i_l)_{i\in K\cup K'} \in p_{K,l}(S) \times (\{0,1\}^l)^{K'} \right\}$$

it follows that $\#p_{K\cup K',l+r}(S) = 2^{(k+k')r+k'l}\#p_{K,l}(S)$, we conclude that the definition of μ is independent from the choice of K and l.

It immediately follows from this definition that

- $\mu(\emptyset) = 0$ and $\mu(\text{Sspace}) = 1$
- If $S \cap S' = \emptyset$ then $\mu(S \cup S') = \mu(S) + \mu(S')$

From these facts, we conclude that extending μ on the σ-algebra generated by $\mathcal{U}$ we obtain a probability measure.

Now that we properly defined the size of a sampling set we can define non negligible sampling families. We start by recalling the usual definition of a negligible function:

Definition 2.10. A function f from $\mathbb{N}$ to $\mathbb{R}$ is *negligible* if for all polynomial p there exists N such that for all $n \geq N$ we have

$$f(n) < \frac{1}{p(n)}$$

The intuition of a non negligible sampling family is a family of sampling sets that do not decrease in size too quickly, which ensures that if something bad happens in a sampling family, it also happens with non negligible probability in the computational model.

Definition 2.11. A *sampling family* is a family $(S_\eta)_{\eta\in\mathbb{N}}$ of uniform sampling sets. It is *non-negligible* if $f : \eta \mapsto \mu(S_\eta)$ is not negligible. We will often use $|S|$ to denote f.

For two sampling families S, S', we write $S \subseteq S'$ if for all η we have $S_\eta \subseteq S'_\eta$.

Kripke worlds and accessibility relation

In all generality, we will consider the class of Kripke domains $\mathcal{D}$ defined by

- A non-negligible sampling family $S = (S_\eta)_{\eta\in\mathbb{N}}$. The worlds of $\mathcal{D}$ are all non negligible sampling families S' such that $S' \subseteq S$. As one could expect the accessibility relation is the $\subset$ relation between sampling families.
- For each η and each constant c a random variable $c_\eta : S_\eta \to \{0,1\}^*$
- For each η and each function f in $\mathcal{F}$ of arity l a random variable

$$f_\eta : S_\eta \to \left((\{0,1\}^*)^l \to \{0,1\}^*\right)$$

- For each predicate p of arity l a random variable

$$p_\eta : S_\eta \to \left((\{0,1\}^*)^l \to \{\top,\bot\}\right)$$

For a term t and a sampling $\tau \in S_\eta$ we write

$$[\![t]\!]_\tau = \begin{cases} f_\eta(\tau)([\![t_1]\!]_\tau, \ldots, [\![t_l]\!]_\tau, \tau_1) \text{ if } t = f(t_1, \ldots, t_n) \\ c_\eta(\tau) \text{ if } t = c \end{cases}$$

Similarly, we define $[\![p(t_1, \ldots, t_l)]\!]_\tau$ as $p_\eta(\tau)([\![t_1]\!]_\tau, \ldots, [\![t_l]\!]_\tau)$.

The reason for which we consider these Kripke structures is that the problem we want to tackle is the existence of a PPT adversary against a protocol. The non-negligible sampling family S will be the set of drawings of both honest

and adversarial randomness on which the adversary succeeds at breaking the security property.

We are left with defining what it means for a predicate to be satisfied in a world S. The intuition is that $p \in \mathcal{P}$ is true on S if it is true with overwhelming probability on S. More formally, we have $S \models p(t_1, \ldots, t_n)$ if and only if the function $\eta \mapsto \mathbb{P}(\tau \in S_\eta \text{ and } [\![p(t_1, \ldots, t_n)]\!]_\tau = \bot)$ is negligible.

Example 2.12. Assume that in the Kripke structure nonces are interpreted as random bitstrings of length η. Let S be a world in this Kripke structure. Let n_1, n_2 be two nonces. Let us consider the binary disequality predicate $\neq$. We have $S \models n_1 \neq n_2$ as $\mathbb{P}(\tau \in S_\eta \text{ and } [\![n_1]\!]_\tau = [\![n_2]\!]_\tau) \leq 2^{-\eta}$

As one could expect, the picture is quite different for the $\triangleright$ predicate. For an integer i and a PPT $\mathcal{A}$, we define $\mathcal{A}_i$ as the PPT $\mathcal{A}$ using the i-th random tape for its randomness. We say that $S \models t_1; \ldots; t_n \triangleright u$ if there exists an index i with $i \in \mathbb{N}$ and a PPT $\mathcal{A}$ such that $\eta \mapsto \mathbb{P}(\tau \in S_\eta \text{ and } \mathcal{A}_i([\![t_1]\!]_\tau, \ldots, [\![t_n]\!]_\tau) \neq [\![u]\!]_\tau)$ is negligible. If $\mathcal{A}_i$ is such a witness, we write $S, \mathcal{A}_i \models t_1; \ldots; t_n \triangleright u$, however, we will often omit the index when it is not relevant to the proof.

Example 2.13. Let $t \in T(\mathcal{F}, \mathcal{C})$. Let us show that for any world S, we have $S \models t \triangleright t$. Let $\mathcal{A}$ be the PPT that simply returns its input, we have (for any index i) $S, \mathcal{A}_i \models t \triangleright t$.

Example 2.14. Assume that the constant b is interpret as a random bit. Let us now consider the world S where for all τ in S, the i-th component of τ starts with $[\![b]\!]_\tau$. It is easy to see that such a sampling family is non-negligible. Let now $\mathcal{A}$ be the PPT that returns the first bit of its random tape. We have $S, \mathcal{A}_i \models \emptyset \triangleright b$.

Note that in section 2.1 we say that we assume the ACIN axioms, we therefore have to prove that these axioms hold in every Kripke world. It is clear for associativity as we are considering sets as lists. For commutativity it is enough to notice that if σ is a permutation of $\{1, \ldots, n\}$, if we have $S, \mathcal{A}_i \models t_1, \ldots, t_n \triangleright u$, defining $\mathcal{A}_i^\sigma(x_1, \ldots, x_n) = \mathcal{A}_i(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ we have $S, \mathcal{A}_i^\sigma \models t_{\sigma(1)}, \ldots, t_{\sigma(n)} \triangleright u$. The same technique holds for idempotence.

We further restrict the class of Kripke models we consider as follows:

- For each function f in $\mathcal{F}$ we fix a probabilistic polynomial time Turing machine $\mathcal{A}^f$, we fix $f_\eta(\tau)(x_1, \ldots, x_l) = \mathcal{A}_1^f(\eta, x_1, \ldots, x_l)$ where η is coded in unary. All functions share the same randomness. This randomness is also accessible to the adversary as it is stored on the first adversarial tape. Therefore if one wants to model private or independent randomness, as in probabilistic encryption, it is necessary to include it explicitly in the arguments of the function. We do not restrict the functions to deterministic functions as the adversarial messages are randomised functions of the adversarial knowledge.
- For each predicate p in $\mathcal{P}$ we fix a polynomial time Turing machine $\mathcal{A}^p$, we fix $p_\eta(\tau)(x_1, \ldots, x_l) = \top$ if and only if $\mathcal{A}_1^p(\eta, x_1, \ldots, x_l)$ accepts.
- For each sort s of constants we fix a PPT $\mathcal{A}^{\mathsf{s}}$. If c is of sort s, we take $c_\eta(\tau) = \mathcal{A}_c^{\mathsf{s}}(\eta)$.

Intuitively, these restrictions ensure that the Kripke models we consider correctly reflect the computational model. We call these models *computational Kripke models*.

Recall that the problem we want to solve is: given a set of modal formulae (we will consider this set to be ground and possibly infinite), is there a computational

Kripke model of these formulae? For simplicity reasons, we do not want to solve the problem within modal logic. We then use the Fitting's embedding of first order logic into S_4 in order to ensure that if such a model exists, then a first order model also exists. Note that the converse might not hold.

The S_4 modal logic has two modalities: $\Box$ and $\Diamond$. The $\Box$ modalities is the "always" modality: $S \models \Box F$ means that for every world S' accessible from S, $S' \models F$. The $\Diamond$ modality is the "eventually" modality: $S \models \Diamond F$ is defined as there exists a world S' accessible from S such that $S' \models F$. Additionally it admits as Kripke models all Kripke structures in which the accessibility relation is reflexive and transitive. Note that this is in particular the case of the computational Kripke models.

Let us give a statement of the Fitting's embedding ([Fit70]).

Theorem 1 (Fitting's embedding)**.** *Let us define A^* for each first order formula A: for A atomic, $A^* = \Box\Diamond A$ and*

$$\begin{array}{rcl}(A \wedge B)^* & = & \Box\Diamond(A^* \wedge B^*)\\ (A \vee B)^* & = & \Box\Diamond(A^* \vee B^*)\\ (A \to B)^* & = & \Box\Diamond(A^* \to B^*)\\ (\neg A)^* & = & \Box\Diamond(\neg A^*)\\ (\forall x.A(x))^* & = & \Box\Diamond(\forall x.A(x)^*)\\ (\exists x.A(x))^* & = & \Box\Diamond(\exists x.A(x)^*)\end{array}$$

Let $\mathcal{S}$ be a set of first order formulae. $\mathcal{S}$ is consistent if and only if $\mathcal{S}^*$ is consistent in S_4.

Given a set of (first order) axioms $\mathcal{A}$ such that $\mathcal{A}^*$ holds in every computational Kripke model, and a set of ground formulae $\mathcal{S}$, we get that if there exists a computational Kripke model in which $\mathcal{S}^*$ holds, then there exists a first order model of $\mathcal{A} \cup \mathcal{S}$. In other words, as $\mathcal{S}$ will encode a trace of the protocol together with the negation of the security property, if the protocol has a computational attack, then $\mathcal{A} \cup \mathcal{S}$ is consistent for the $\mathcal{S}$ corresponding to the attack trace.

Remark 2.15. The converse might not be true. Recall that we are not considering all Kripke models of S_4, only the computational Kripke models. Hence $\mathcal{A} \cup \mathcal{S}$ could be consistent, while its model does not show an attack (though we do not have an example of such a false attack).

2.3 The execution model

In order to state the computational soundness theorem, we need to define properly the execution model and the security definitions. This execution model is very permissive, we define protocols as transition system and do not stick to any particular syntax or semantics: the operational semantics of process calculi such as applied π-calculus or the transition systems specified by multiset rewrite rules can be easily formulated using the next definition.

As in [BC12] we define a protocol as follows, assuming Q is a (possibly infinite) set of control states with a distinguished initial state q_0:

Definition 2.16. A protocol is a recursive set of tuples

$$(q(\overline{n}), q'(\overline{n}.\overline{n'}), \langle x_1, \ldots, x_k\rangle, x, \psi, s)$$

where $q, q' \in Q, x_1, \ldots, x_k, x$ are variables $\overline{n}, \overline{n'}$ are finite sequences of names, ψ is a first order formula over the set of predicate symbols $\mathcal{P}$ and function symbols $\mathcal{F}$ and the names $\overline{n} \cup \overline{n'}$, whose free variables are in $\{x_1, \ldots, x_n, x\}$ and s is a term whose free variables are in $\{x_1, \ldots, x_n, x\}$.

Let us give some intuition. The formula ψ is the guard of the transition, enforced by the protocol. It typically expresses either the verification that a message is well formed (decrypts or projects correctly) or that some equalities hold. A transition from q to q' is possible, if given the previous inputs $x_1, \ldots, x_n$ and the last input x, the guard ψ holds. If the transition is fired, the names $\overline{n'}$ are generated and the message s is sent out.

Example 2.17. Let us consider the following simple one-way authentication protocol:

$$\begin{array}{lcll} A & \rightarrow & B: & \{n, A\}_{pk_B} \\ B & \rightarrow & A: & \{n\}_{pk_A} \end{array}$$

We assume, without loss of generality that the action of A is triggered by the adversary with some signal. For one instance of each role, the transition system is as follows:

- the transition $(q_0(), q_A(n.r), \langle\rangle, x_A, \top, \{n, A\}^r_{pk_B})$ for the action of A
- let us call v_n^{AB} the term $\mathrm{adec}(x_{AB}, sk_B)$; if A has already played, the action of B yields the transition

$$(q_A(n.r), q_{AB}(n.r.r'), \langle x_A\rangle, x_{AB}, (\pi_2(v_n^{AB}) = A), \{\pi_1(v_n^{AB})\}^{r'}_{pk_A})$$

- let us call v_n^B the term $\mathrm{adec}(x_B, sk_B)$; if B plays first, we have the transition

$$(q_0(), q_B(r')\langle\rangle, x_B, (\pi_2(v_n^B) = A), \{\pi_1(v_n^B)\}^{r'}_{pk_A})$$

- finally $(q_B(r'), q_{BA}(r'.n.r), \langle x_B\rangle, x_{BA}, \top, \{n, A\}^r_{pk_B})$ for the subsequent action of A.

Note that this transition systems does not consider any concurrency, there is only one adversary, one channel. This does mean that some translation work has to be done from usual protocol models such as applied π-calculus, mainly unravelling all interleavings as can be seen in the previous example.

Having defined our protocol model, we can now define security properties as recursive functions from the state to the formulae. The intuition is that the security property represents the set of conditions that should be verified in a state. Typically it will be stating that some constant should not be computable with the messages that were sent on the network, or that some equalities should hold (in other terms some parties should agree on some values) if the protocol reaches some particular "accepting" state.

Definition 2.18 (Security property)**.** A *security property* is a recursive function from Q to the first order formulae over $\mathcal{P}$ and $\mathcal{F}$.

Example 2.19. Let us come back to example 2.17. We wish to express that the nonce n is not computable from the messages that are sent. This is expressed with the following property p:

- $p(q_0) = \neg\emptyset \triangleright n$
- $p(q_A) = \neg[\{n, A\}^r_{pk_B} \triangleright n]$ and $p(q_{AB}) = \neg[\{n, A\}^r_{pk_B}; \{\pi_1(v_n^{AB})\}^{r'}_{pk_A} \triangleright n]$
- $p(q_B) = \neg[\{\pi_1(v_n^B)\}^{r'}_{pk_A} \triangleright n]$ and $p(q_{BA}) = \neg[\{\pi_1(v_n^B)\}^{r'}_{pk_A}; \{n, A\}^r_{pk_B} \triangleright n]$

Expressing agreement is even simpler. We need to state that if B accepts a nonce, then it should be the nonce sent by A. If B has not accepted yet (in q_0, q_A) the security property is $\top$. In every other state, it is $\pi_1(v_n^B) = n$.

We now need to give a precise semantics for this model. Given a security parameter η, computational interpretation of functions and predicates, the computational semantics behaves as follows. Note that we do need an adversary to compute and schedule the communications.

Definition 2.20 (Computational semantics)**.** Given a protocol P and a PPT adversary $\mathcal{A}$, we define the execution of P with $\mathcal{A}$ as environment (written $P \| \mathcal{A}$). As a lot of computations will be probabilistic, we fix τ a random sampling and let r be the first adversarial randomness tape in τ.

- We define a *computational state* as $(q(\overline{n}), \langle h_1, \ldots, h_n \rangle, \langle m_1, \ldots, m_l \rangle)$ where $q \in Q$, $h_1, \ldots, h_n$ are bitstrings standing for the previously received message and $m_1, \ldots, m_l$ are the messages sent over the network so far.
- We define the transition relation $\rightarrow_\tau$ as follows:

$$(q(\overline{n}), \langle h_1, \ldots, h_n \rangle, \langle m_1, \ldots, m_l \rangle) \rightarrow_\tau \\ (q'(\overline{n.\overline{n'}}), \langle h_1, \ldots, h_n, h \rangle, \langle m_1, \ldots, m_l, m \rangle)$$

  if $(q(\overline{n}), q(\overline{n.\overline{n'}}), \langle x_1, \ldots, x_k \rangle, x, \psi, s) \in P$ with

  1. $h = \mathcal{A}_r(\langle m_1, \ldots, m_l \rangle)$ is the new message computed by the adversary
  2. $\tau, \{x_1 \mapsto h_1, \ldots, x_n \mapsto h_n, x \mapsto h\} \models \psi$
  3. $m = [\![s]\!]_{\tau,\sigma}$ with $\sigma = \{x_1 \mapsto h_1, \ldots, x_n \mapsto h_n, x \mapsto h\}$ i.e. the interpretation of all constants are taken according to τ and the variables are replaced by the corresponding adversarial messages.

A computational transition sequence starting from state the initial state q_0 is called a *computational trace*.

Note that with any reasonable translation of protocols into this model we just redefined the usual computational semantics.

Example 2.21. Let us come back to example 2.17. Let us describe the honest execution of the protocol where A plays first then B and the message is simply forwarded by the adversary. Let τ be a sampling. We get the following computational trace:

$$(q_0(), \langle\rangle, \langle\rangle) \rightarrow_\tau (q_A(n.r), \langle h_1 \rangle, \langle [\![\{n, A\}^r_{pk_B}]\!]_\tau \rangle) \\ \rightarrow_\tau (q_{AB}(n.r.r'), \langle h_1, [\![\{n, A\}^r_{pk_B}]\!]_\tau \rangle, \langle [\![\{n, A\}^r_{pk_B}]\!]_\tau, [\![\{n\}^{r'}_{pk_A}]\!]_\tau \rangle)$$

We say as usual that a protocol P satisfies the security property p if for any PPT $\mathcal{A}$, the following function f is negligible

$$f : \eta \mapsto \mathbb{P}_\tau \big[P \| \mathcal{A} \rightarrow^*_\tau (q(\overline{n}), \langle h_1, \ldots, h_n \rangle, \langle m_1, \ldots, m_l \rangle) \\ \text{with } \tau, \{x_1 \mapsto h_1, \ldots, x_n \mapsto h_n)\} \models \neg p(q) \big]$$

In other terms, the protocol P satisfies the security property p if and only if any polynomial time attacker computing the messages sent to the protocol, a state of P where p is not satisfied can only be reached with negligible probability.

While this computational semantics is basically the usual semantics of the computational model we need to abstract it into a symbolic semantics. Indeed if we do not fix one particular sampling, it is infinitely branching. Moreover, we do not want to consider one adversary explicitly, as quantifying over all PPT is not really an option if we want to have an abstract model. What we do in the symbolic semantics, as in [BC12], is to regroup all traces that follow the same control path. In other terms, we give a symbolic semantics that, at each step, gather all the traces that satisfy one transition guard ψ. Afterwards, the Kripke semantics we defined in the previous section will allow us to reason on whether or not there exists an adversary capable of executing such a symbolic trace. In order to define a symbolic semantics, we need to define functions to represent the adversarial messages. We therefore assume that $\mathcal{F}$ contains $\{\text{handle}_i\,/i|i \in \mathbb{N}\}$.

Definition 2.22 (Symbolic semantics)**.** A *symbolic state* of the network consists of:

- a control state $q \in Q$ together with a sequence of names (that have been generated so far) $n_1, \ldots, n_k$
- a sequence of ground terms $h_1, \ldots, h_n$ recording the inputs of the protocol
- a ground extended term ϕ called the *frame* (recording the agents outputs)
- a set of ground formulae Θ built over $\mathcal{P}, =, \triangleright$ (the conditions that have to be satisfied in order to reach the state).

A symbolic transition of a protocol P is

$$(q(\overline{n}), \langle h_1, \ldots, h_m\rangle, \phi, \Theta) \to (q'(\overline{n'}), \langle h_1, \ldots, h_m, h\rangle, \phi; s, \Theta \cup \Psi)$$

if there exists a transition rule

$$(q(\overline{n}), q'(\overline{n'}), \langle x_1, \ldots, x_m\rangle, x, \psi, s') \in P$$

with

- $s = s'\{x_1 \mapsto h_1, \ldots, x_i \mapsto h_i\}$
- $h = \text{handle}_i(\phi)$
- $\Psi = \phi \triangleright h \ \wedge \ \psi\{x_1 \mapsto h_1, \ldots, x_i \mapsto h_i\}$

A symbolic transition sequence starting from the initial state q_0 is called a *symbolic trace*.

Note that there is exactly one symbolic transition per rule of the protocol, therefore we can map each computational trace to the symbolic trace executing the same protocol rules.

Example 2.23. Let us come back to example 2.17. Recall the computational trace we described:

$$\begin{aligned}(q_0(), \langle\rangle, \langle\rangle) &\to_\tau (q_A(n.r), \langle h_1\rangle, \langle [\![\{n, A\}^r_{pk_B}]\!]_\tau\rangle)\\ &\to_\tau (q_{AB}(n.r.r'), \langle h_1, [\![\{n, A\}^r_{pk_B}]\!]_\tau\rangle, \langle [\![\{n, A\}^r_{pk_B}]\!]_\tau, [\![\{n\}^{r'}_{pk_A}]\!]_\tau\rangle)\end{aligned}$$

This computational execution is mimicked by the following symbolic trace:

$$(q_0(), \langle\rangle, \emptyset, \emptyset) \ \to \ (q_A(n.r), \langle h_1\rangle, \phi_1, \Theta_1) \ \to \ (q_{AB}(n.r.r'), \langle h_1, h_2\rangle, \phi_2, \Theta_2)$$

with the frames $\phi_1 = \{n, A\}^r_{pk_B}$ and $\phi_2 = \phi_1; \{\pi_1(\text{adec}(h_2, sk_B))\}^{r'}_{pk_A}$. The protocol guards that are checked along the trace, together with the computability of adversarial messages are:

- $\Theta_1 = \emptyset \triangleright h_1$
- $\Theta_2 = \Theta_1, \quad \{n, A\}^r_{pk_B} \triangleright h_2, \quad \pi_2(\text{adec}(h_2, sk_B)) = A$

Note that the symbolic execution represents any computational trace following the same control path as the one of the original computational trace. Therefore it represents infinitely many computational transition sequences at once.

Definition 2.24. Let $\mathcal{A}$ be a set of first order axioms. We say that a symbolic trace

$$(q_0(\overline{n_0}), \emptyset, \phi_0, \emptyset) \to \cdots \to (q_m(\overline{n_m}), \langle h_1, \ldots, h_m\rangle, \phi_m, \Theta_m)$$

of P is *valid* with respect to $\mathcal{A}$ if for $0 \leq i \leq m$ the set of formulae $\Theta_i, \mathcal{A}$ is satisfiable.

Note that as $(\Theta_i)_{i\leq m}$ is an increasing family of sets, it is enough to check the condition on Θ_m.

Now that the symbolic transition system is defined, and that we have a notion of which traces are valid, we can move on to the definition of symbolic security. Intuitively, a protocol is symbolically insecure if and only if there exists first order model $\mathcal{M}$ that is a witness to the validity of a trace t and $\mathcal{M} \models \neg p(t)$.

Definition 2.25 (symbolic security)**.** Let P be a protocol, p be a security property. We say that P is *symbolically secure* with respect to axioms $\mathcal{A}$ and property p if for every valid symbolic transition sequence of P with respect to $\mathcal{A}$

$$(q_0(\overline{n_0}), \emptyset, \phi_0, \emptyset) \to \cdots \to (q_m(\overline{n_m}), \langle h_1, \ldots, h_m\rangle, \phi_m, \Theta_m)$$

the negation of the security property is incompatible with the axioms: i.e. the set of formulae $\mathcal{A}, \Theta_m, \ \neg(p(q_m(\overline{n_m}))\{x_1 \mapsto h_1, \ldots, x_n \mapsto h_n\})$ is inconsistent.

The whole point of considering this symbolic model and defining the computational Kripke models is to define a symbolic model that is as permissive as the computational model. In order to achieve this goal we have to relate the axioms used in the previous definitions to the computational model.

Definition 2.26. Let $\mathcal{A}$ be a set of first order formulae. We say that $\mathcal{A}$ is computationally sound if for every computational Kripke model $\mathcal{M}$ we have $\mathcal{M} \models \mathcal{A}^*$.

Having stated computational soundness for the axioms, we can state the main theorem. This theorem simply states that, as long as the axioms we consider are computationally sound, checking symbolic security is sufficient to obtain computational security.

Theorem 2 (Computational soundness)**.** *Assume that P has a finite number of symbolic traces. Let $\mathcal{A}$ be a set of computationally sound first order axioms. If the protocol P is symbolically secure with respect to axioms $\mathcal{A}$ and security property p then it is computationally secure with respect to security property p.*

Proof. Let us assume that P is computationally insecure. Let us consider an adversary $\mathcal{A}$ breaking the computational security of P. There is a non negligible sampling family S such that for all η and all $\tau \in S_\eta$ we have

$$P\|\mathcal{A} \to_\tau (q(\overline{n}), \langle h_1, \ldots, h_n\rangle, \langle m_1, \ldots, m_l\rangle) \quad \text{with } \tau, \{x_1 \mapsto h_1, \ldots, x_n \mapsto h_n\} \models \neg p(q)$$

Each of these attack traces can be mapped to a symbolic trace of P. As P only has a finite number of possible symbolic traces we get a non-negligible sampling family S' and a symbolic trace $t = (q_0(\_), \_, \_, \_) \rightarrow^* (q_n(\overline{n}), \langle h_1, \ldots, h_n\rangle, \phi_n, \Theta_n)$ of P such that for every τ in S' there exists a computational trace t_τ

$$P \| \mathcal{A} \rightarrow_\tau (q(\overline{n}), \langle h_1^\tau, \ldots, h_n^\tau\rangle, \langle m_1^\tau, \ldots, m_n^\tau\rangle) \quad \text{with } \tau, \{x_1 \mapsto h_1^\tau, \ldots, x_n \mapsto h_n^\tau\} \models \neg p(q)$$

which is abstracted by t. Let us write $(q_i(\_), \langle h_1^\tau, \ldots, h_i^\tau\rangle, \langle m_1^\tau, \ldots, m_i^\tau\rangle)$ the i-th step of t_τ. For $0 < i \leq n$, let $(q_{i-1}(\_), q_i(\_), \langle x_1, \ldots, x_{i-1}\rangle, x_i, \psi_i, s_i)$ the rule of P used to fire the i-th transition of t_τ. Note that as we assume that all the t_τ are abstracted by the same symbolic trace, this definition does not depend on τ.

Let $h_1, \ldots, h_n$ be the terms representing the protocol's inputs in the symbolic trace t. We define σ as the substitution $\{x_1 \mapsto h_1, \ldots, x_n \mapsto h_n\}$ and σ_τ as $\{x_1 \mapsto h_1^\tau, \ldots, x_n \mapsto h_n^\tau\}$. Let us now show that $S' \models [\Theta_n, \neg(p(q)\sigma)]^*$. The only functions whose interpretations are not fixed are the handles. We simply interpret the $\text{handle}_i(x_1, \ldots, x_i)$ function as $\mathcal{A}(x_1, \ldots, x_i)$. It is now enough to remark that S' actually is computational Kripke model of $[\Theta_m,\ \neg p(q)\sigma]^*$. Note that $\mathcal{A}$ is a witness for all the $\triangleright$ formulae and all the other formulae hold on S' by construction. Let us state this formally.

Note that as all the constants are drawn according to τ in t_τ, with our definition of the handles we get $[\![\phi_i]\!]_\tau = \langle m_1^\tau, \ldots, m_i^\tau\rangle$. We consider ψ a formula in Θ_n, let us assume the $\psi \in \Theta_i \backslash \Theta_{i-1}$. Then we have

- either $\psi = \phi_{i-1} \triangleright h_i$, and then (writing the first adversarial random tape as r) $S', \mathcal{A}_1 \models \phi_{i-1} \triangleright h_i$ by definition of the interpretation of handle_i. Now we observe that for any $S'' \subseteq S'$, we have $S'', \mathcal{A}_1 \models \phi_{i-1} \triangleright h_i$. We conclude that $S' \models \Box\Diamond\phi_{i-1} \triangleright h_i$, as for all successor S'' of S' there exists a successor of S'' on which $\phi_{i-1} \triangleright h_i$ holds (in this particular case S'' itself).
- or $\psi = \psi_i$ and for all $\tau \in S'$ we have $\tau, \sigma_\tau \models \psi_i$ as this is exactly the computational requirement for the protocol to execute trace t. Therefore as in the previous point $S' \models (\psi_i\sigma)^*$

The only case left out is the case of $\neg p(q)\sigma$. As previously, we have for all $\tau \in S'$, $\tau, \sigma_\tau \models \neg p(q)$ hence $S' \models (\neg p(q)\sigma)^*$

Now remark that S' is also a computational Kripke model of $\mathcal{A}^*$ as the axioms are assumed computationally sound. Fitting's embedding yields the first order satisfiability of $\Theta_n,\ \neg(p(q)\sigma),\ \mathcal{A}$. This concludes the proof. □

The previous soundness theorem can be seen as a trace mapping result. Note that the number of hypotheses involved in the proof and the length of the proof is considerably shorter than usual trace mapping results found in computational soundness proofs (for examples of such proofs see [BP04, MW04, CC08],...). The main reason of this simplicity is that the attacker's possibilities are naturally reflected in our symbolic models, thus we do not have to reason on a way to restrict the meaningful actions of a computational adversary. The assumption that the protocol only has a bounded number of traces is a design choice of this model. If we do not make this hypothesis, we need to consider attacks whose length depending on the security parameter. Considering such attacks would necessitate a representation of this security parameter somewhere in the

symbolic model. In order to keep the model simple, the authors decided in [BC12] to disregard such attacks. We follow that choice here.

2.4 Conclusion

We have defined here, along the ideas of [BC12] albeit in a different formalism originally introduced in [BHO13] a symbolic framework for proving cryptographic protocols. The main idea behind this framework is to define a symbolic attacker that is as powerful as possible. We defined a computational logic reflecting the capabilities of a computational adversary. Using this logic we can formulate a symbolic transition system that is sound with respect to the computational model. The last part is to notice that, thanks to Fitting's embedding of first order logic into S_4, checking whether there exists a valid symbolic trace contradicting the security property can be reduced to proving that a set of first order formulae is satisfiable.

Note that one of the crucial parts is to formulate a set of axioms $\mathcal{A}$ in first order logic that are computationally sound (i.e. that hold in any computational Kripke model). This is the problem we tackle in the next chapter. The reason for which this part of formulating axioms is so crucial is because without any such axiom, the interpretation of the predicates (and in particular the $\triangleright$) is not restricted in any way when checking first order satisfiability. However in the computational model, the functions (in particular the cryptographic primitives) are instantiated and therefore have a lot of properties. These properties formulated as axioms naturally hold in any computational Kripke model.

Chapter 3

Axioms and soundness proofs

In the previous chapter, we showed that proving a protocol computationally secure amounts to checking the consistency of a set of ground formulae $\mathcal{S}$ together with a set of computationally sound axioms $\mathcal{A}$. In this chapter, we propose a list of computationally sound axioms modelling the adversarial capabilities. We start by presenting what we call the "core axioms" that restrict the capabilities of any PPT adversary independently of any hypotheses on the cryptographic primitives. In other words, these core axioms represent the restriction of an adversary that hold even if the cryptographic primitives are insecure. These axioms are proven sound with very minimal hypotheses on the implementation of functions. We then proceed to model axioms for classical properties of encryption schemes, namely integrity and secrecy.

Apart from a slight generalisation, the core axioms are the ones proven sound in [BC12, BAS12]. However, the formal model we consider here is not exactly the same as the one presented in [BC12]. The proofs mostly rely on the same ideas, but are presented in a different setting. The cryptographic axioms however have some differences. We identify a missing cryptographic hypothesis for the proof of the secrecy axiom in the symmetric case. The original proof from Bana and Comon, does not carry over the symmetric encryption case. We show here that this hypothesis is necessary for the axiom to be sound. The main contribution however is allowing for a less restrictive use of (symmetric) keys in the axioms, which is necessary in order to prove lots of protocols involving symmetric encryption, in particular key-exchange protocols. The third contribution regarding the cryptographic axioms is the design of an axiom for unforgeability of encryption with a very different approach from [BHO13].

3.1 Core axioms

Let us first present the core axioms, that are sound regardless of the cryptographic properties. These axioms represent the capabilities of an adversary that are independent of any security hypothesis on the cryptographic primitives or implementation hypothesis on the functions. The only assumption we make is that all functions are implemented by polynomial time algorithms which is

in practice always satisfied as the protocols using these functions are always supposed to run in polynomial time. We also give a precise computational condition under which random values such as nonces and keys are not guessable and provide with the corresponding axiom. All of the axioms presented in this section were originally introduced in [BC12]. The proofs are however presented in the formalism proposed in chapter 2 instead of the original formalism from [BC12]. Moreover we extend the freshness axiom by weakening the computational hypotheses.

One interesting point is that they are very similar to the way we would model the deductive capabilities of a Dolev Yao adversary. This makes sense because, in the Dolev Yao world, as all primitives are considered "perfect", the adversary possesses only these minimal capabilities. In the present context however, they are just a lower bound on the capabilities of an adversary. Although stating such capabilities is not necessary, it makes life simpler;other axioms can be stated in a simpler way, as shown by the following example.

Example 3.1. Let us consider the following statement: $\langle \{n\}_k^r, A\rangle \triangleright n$. It should not be satisfied by any reasonable encryption scheme.

We are left with two choices for designing the axioms.

- We can state that the pair is a computable function (in other words $x; y \triangleright \langle x, y\rangle$) therefore obtaining $\{n\}_k^r; A \triangleright n$. In addition we design an axiom stating that n can not be computed from $\{n\}_k^r$: $\{n\}_k^r \not\triangleright n$.
- We can omit the ability of the adversary to compute pairs and design an axiom such as $\langle \{n\}_k^r, A\rangle \not\triangleright n$.

We chose the first option. The second option would need to design a much more involved secrecy axiom whose proof would be much more technical.

3.1.1 Reflexivity, transitivity and monotonicity

The three first axioms we describe here are the only axioms in this framework that are sound irrespectively of any other computational hypothesis. They reflect some properties of Turing machines.

The simplest axiom here is the reflexivity (denoted by Refl) axiom, it basically states that the identity function is computable.

$$\boxed{X; x \triangleright x}$$

The first non-trivial axiom is the transitivity axiom (denoted by Tr), intuitively it states that giving to the adversary knowledge that it can already compute does not increase its capabilities. In other words, it states that the composition of two PPT is also a PPT.

$$\boxed{X \triangleright x \quad , \quad X; x \triangleright y \quad \rightarrow \quad X \triangleright y}$$

The third core axiom is the monotonicity axiom (denoted by Mon) that states that the adversary may discard some of its inputs. In terms of computability, it means that if a n-ary function f is computable then so is $g : (x_1, \ldots, x_{n+1}) \mapsto f(x_1, \ldots, x_n)$

$$\boxed{X \triangleright y \quad \rightarrow \quad X; x \triangleright y}$$

Even if these three axioms seem fairly natural, due to the relative complexity of the underlying modal logic and the Fitting's translation, we still need to prove

their soundness. We also hope that the proof of the next proposition will be a good example that very natural axioms have a very natural soundness proof.

Proposition 3.2. *The axioms* Refl, Tr *and* Mon *are computationally sound irrespectively of any computational hypothesis.*

Proof. Let us start with the reflexivity axiom, Fitting's translation yields:

$$\Box\Diamond X; x \triangleright x$$

Let S be a sampling family, let $\mathcal{A}_\pi$ be the (polynomial time) Turing machine returning its last argument (i.e. $\mathcal{A}_\pi(X, w) = w$). We have

$$S, \mathcal{A}_\pi \models X; x \triangleright x$$

We now prove the monotonicity axiom, which is translated as:

$$\Box\Diamond(\Box\Diamond X \triangleright y \quad \rightarrow \quad \Box\Diamond X; x \triangleright y)$$

We consider an instance of this axiom:

$$\Box\Diamond(\Box\Diamond t_1; \ldots; t_n \triangleright u \quad \rightarrow \quad \Box\Diamond t_1; \ldots; t_n; t' \triangleright u)$$

Let S be a sampling in which there exists a PPT $\mathcal{A}$ such that $\mathcal{A}, S \models X \triangleright y$. Let now $\mathcal{A}'$ be the PPT that takes $n+1$ arguments, ignores the last one and computes as $\mathcal{A}$. Note that $\mathcal{A}'([\![t_1]\!], \ldots, [\![t_n]\!], [\![t']\!])$ if and only if $\mathcal{A}([\![t_1]\!], \ldots, [\![t_n]\!])$. We can conclude that for any sampling family S

$$S \models \Box\Diamond t_1; \ldots; t_n \triangleright u \quad \rightarrow \quad \Box\Diamond t_1; \ldots; t_n; t' \triangleright u$$

We now consider an instance of the transitivity axiom, more precisely:

$$\Box\Diamond(\Box\Diamond(\Box\Diamond t_1; \ldots; t_n \triangleright t \wedge \Box\Diamond t_1; \ldots; t_n; t \triangleright u) \quad \rightarrow \quad \Box\Diamond t_1; \ldots; t_n \triangleright u)$$

Let S be a sampling family, assume the left hand side of the implication holds, let us take $S_1 \subseteq S$, we want to prove that there exists $S_2 \subseteq S_1$ and a PPT $\mathcal{A}$ for which $S_2, \mathcal{A} \models t_1; \ldots; t_n \triangleright u$.

1. According to the left hand side of the implication, there exists $S_1' \subseteq S_1$ such that $S_1' \models \Box\Diamond t_1; \ldots; t_n \triangleright t \wedge \Box\Diamond t_1; \ldots; t_n; t \triangleright u$
2. $\Box\Diamond t_1; \ldots; t_n \triangleright t$ yields $S_1'' \subseteq S_1'$ and a PPT $\mathcal{B}$ such that $S_1'', \mathcal{B} \models t_1; \ldots; t_n \triangleright t$. Let us remark that for every sample family U smaller than S_1'', we have $U, \mathcal{B} \models t_1; \ldots; t_n \triangleright t$
3. $\Box\Diamond t_1; \ldots; t_n; t \triangleright u$ gives $S_2 \subseteq S_1''$ such that $S_2, \mathcal{B}' \models t_1; \ldots; t_n; t \triangleright u$

Summing everything up, taking $\mathcal{A}(x_1, \ldots, x_n) = \mathcal{B}'(x_1, \ldots, x_n, \mathcal{B}(x_1, \ldots, x_n))$ we get $S_2, \mathcal{A} \models t_1; \ldots; t_n \triangleright u$ □

Having proven the axioms that do not rely on any hypotheses, we can move on to the other core axioms relying on natural assumptions.

3.1.2 Functionality

We now assume that all functions in $\mathcal{F}$ are computable in polynomial time. The functionality axioms simply state that fact. This set of axioms is quite easy to prove, as the witness for the computability of a function in $\mathcal{F}$ is precisely the machine computing this function.

For f in $\mathcal{F}$, we define the functionality axiom ($\textsc{Fun}_f$) as follows,

$$\boxed{x_1; \ldots; x_n \triangleright f(x_1, \ldots, x_n)}$$

Proposition 3.3. *If f is computable in* **PTIME***, then* $\textsc{Fun}_f$ *is computationally sound.*

Proof. Let A_f be the polynomial time Turing machine computing f. Let us consider an instance of the $\textsc{Fun}_f$ axiom:

$$\Box\Diamond t_1; \ldots; t_n \triangleright f(t_1, \ldots, t_n)$$

To obtain the soundness of this axiom, we only need to observe that, for any sample family S,

$$S, A_f \models t_1; \ldots; t_n \triangleright f(t_1, \ldots, t_n)$$

□

Do note that if one wants to add some function that if not computable in **PTIME** for technical reasons (for example for encoding practically uncheckable conditions in the trace), it is enough to exclude it from the functions for which we assume the functionality axiom. Therefore the restriction of *all* functions being computable in **PTIME** could be dropped to the cost of having a set of functions that is computable in polynomial time and another that is not. However, as we do not have realistic examples of the usefulness of such non **PTIME** computable functions we decided to give a simpler definition of the set of functionality axioms.

3.1.3 Freshness

In order to actually prove protocols secure, we need to be able to derive contradictions. This can not be done using only the previous axioms. Indeed none of the previous axioms has an empty conclusion. The aim of the freshness axiom is to provide us with the first impossibility in this model. Intuitively, it states that if the distribution of the values of a sort `s` is computationally close enough to the uniform distribution, it should be computationally hard to guess a particular element of this sort. We start by defining formally what we mean by being close enough to the uniform distribution. In [BC12] the freshness axiom is limited to nonces that are interpreted as random bitstrings of length proportional to η. Here we extend the axiom to any sort whose interpretation has enough entropy. Note that, as in a computational Kripke model all constants of a particular sort are interpreted by the same PPT, either all constants of sort `s` have high entropy or none do.

Definition 3.4 (Computationally unpredictable sort)**.** A sort s is said to be *computationally unpredictable* if for any polynomial time Turing machine $\mathcal{A}$ and constant c of sort s, we have

$$\mathbb{P}_\tau\left[[\![c]\!]^\eta_\tau = \mathcal{A}(1^\eta)\right] = \mathrm{negl}(\eta)$$

where negl is a negligible function.

The nonces that are typically interpreted as (possibly tagged) random bitstrings are computationally unpredictable. Note that if an encryption scheme is secure for any usual notion of security, the keys also are unpredictable. Otherwise the adversary would have a good chance of guessing the correct key, thus breaking the scheme.

In order to use this definition in axioms, we need to be able to state in which context (aside from the empty one) an unpredictable constant n is actually unpredictable. A sufficient condition is the absence of occurrence of n in the context.

Definition 3.5 (Freshness)**.** Let t be a term and c be a constant in $\mathcal{C}$. We say that c is *fresh* in t (written $\mathrm{fresh}(c,t)$) if and only if c does not appear as a subterm of t.

By extension, for a set S and a constant c we say that c is *fresh* in S ($\mathrm{fresh}(n,S)$) if and only if for every t in S we have $\mathrm{fresh}(c,t)$.

Now intuitively, if a constant c is fresh in S, and if its sort s is unpredictable, then it should be impossible to guess c from S. We then state the following axiom $\textsc{Fresh}^c_\mathsf{s}$ for every computationally unpredictable sort s and every constant c of sort s.

$$\boxed{\neg X \triangleright c \parallel \mathrm{fresh}(c,X)}$$

We often write Fresh in the following chapters to denote the set of all sound freshness axioms $\{\textsc{Fresh}^c_\mathsf{s} \mid \mathsf{s} \text{ computationally unpredictable and } \mathrm{sort}(c) = \mathsf{s}\}$.

Proposition 3.6. *If s is computationally unpredictable and* $\mathrm{sort}(c) = \mathsf{s}$ *then* $\textsc{Fresh}^c_\mathsf{s}$ *is computationally sound.*

Proof. Let $\neg t_1; \ldots; t_n \triangleright c$ be a valid instance of the $\textsc{Fresh}^c_\mathsf{s}$ axiom scheme. Let S be a sampling family. Note that c is generated independently from the t_i. Let us assume $\mathcal{A}, S \models t_1; \ldots; t_n \triangleright c$.

Let us fix the adversarial random tape and the random tapes of all constants other that c. Applying definition 3.4 yields

$$\mathbb{P}\left[[\![c]\!] = \mathcal{A}([\![t_1]\!], \ldots, [\![t_n]\!])\right] = \mathrm{negl}(\eta)$$

as $\mathcal{A}$ can be considered as a deterministic Turing machine running in polynomial time with respect to η. It follows that S is of negligible size which is a contradiction.

Therefore, we can deduce that for every sampling family S, and any PPT $\mathcal{A}$, $\mathcal{A}, S \not\models t_1; \ldots; t_n \triangleright c$ which concludes the proof. □

Note that this freshness axiom is a generalisation of the freshness axiom given in [BC12]. The original freshness axiom was only stated (and proven) for nonces implemented as random bitstrings of size η. Here in contrast, we have the precise condition under which a constant may not be guessed, allowing our axiom to be used for tagged bitstring but also keys.

3.2 Cryptographic axioms

We have now proven all the core axioms and would be able to verify any protocol that does not use cryptographic primitives. However, our goal is verifying cryptographic protocols. We need to design axioms for cryptographic properties.

In this section, we give axioms for the two most common expected properties of encryption schemes: secrecy and integrity. Note that integrity is formulated differently if we are dealing with asymmetric encryption, in which case it is more precisely defined as a non-malleability property, or if we are considering symmetric encryption where it is usually achieved via unforgeability.

The main contribution here compared to [BC12, BAS12] is the design of syntactic constraints stating in which context a key is usable for various cryptographic security properties. This applies to both symmetric and asymmetric encryption. In [BHO13] the authors developed simultaneously another approach to tackle the same problem of defining key usability, however they take a very different approach, defining a new predicate (quite similar to the $\triangleright$ predicate) for key compromise. Their approach is quite convenient for doing proofs by hand, however considering that our aim is a decision procedure, we believe that adding another variadic predicate with complex interactions with the $\triangleright$ predicate would be much more costly than handling relatively simple syntactic constraint.

We also change the specification of the non-malleability axiom from [BAS12] giving an axiom requiring less assumptions and design an axiom reflecting unforgeability. We also fix a gap in the proofs of the axioms (in the symmetric case) that were proven sound without the which-key concealing hypothesis, while we show here that this assumption is necessary.

We start by recalling the cryptographic properties we use, namely IND-CPA, IND-CCA, KDM and which key concealing and give a short proof that these notions imply that keys should be computationally unpredictable (see definition 3.4). Having defined these notions, we move on to defining the key usability notion, which follows quite closely the power of the adversary in the various definitions. We then proceed to actually prove the axioms.

3.2.1 Cryptographic properties

Let us first present the cryptographic properties for which we devised axioms, by order of strength. We present here the definitions with left or right oracles instead of definitions using real or random oracles, as using left or right oracles is simpler in our proofs. The two definitions are equivalent.

We first give the definition of IND-CPA (originally defined in [GM84]). The intuition behind this definition is that the encryption scheme should not leak any information on the plaintext as long as the adversary is only given access to encryption oracles.

Definition 3.7 (IND-CPA)**.** Let $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme. Let $\mathcal{O}_s(1^\eta)$ (with $s = l$ or $s = r$) be an oracle behaving as follows:

- Generate a key pair $(pk, sk) = \mathcal{K}(1^\eta)$, if the encryption scheme is a symmetric encryption scheme, we assume $pk = sk$ in order to have uniform notation for both cases.
- If the encryption scheme is an asymmetric encryption scheme, on input `key`, return pk.

- On input `encrypt(`m_l`,`m_r`)`, draw a new randomness $r \in \{0,1\}^\eta$ and output $\mathcal{E}(m_s, pk, r)$.

We say that $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ in *IND-CPA secure* if for any PPT $\mathcal{A}$ with one oracle we have

$$\left|\mathbb{P}\left[\mathcal{A}^{\mathcal{O}_l(1^\eta)}(1^\eta) = 1\right] - \mathbb{P}\left[\mathcal{A}^{\mathcal{O}_r(1^\eta)}(1^\eta) = 1\right]\right| = \mathrm{negl}(\eta)$$

where $\mathrm{negl}(\eta)$ is a negligible function in η.

The definition of IND-CCA2 security (originally defined in [RS91]) is much stronger than the definition of IND-CPA as the adversary is also able to decrypt arbitrary messages. In particular this means that no adversary should be able to forge an encryption that is meaningfully related (but not equal) to an honest encryption, in other words non-malleability.

Definition 3.8 (IND-CCA2)**.** Let $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme. Let $\mathcal{O}_s(1^\eta)$ (with $s = l$ or $s = r$) be an oracle behaving as follows:

- Generate a key pair $(pk, sk) = \mathcal{K}(1^\eta)$, if the encryption scheme is a symmetric encryption scheme, assume $pk = sk$ in order to have uniform notation for both cases.
- Initialise a list $L :=$ `[]` that memorises queried encryptions.
- If the encryption scheme is an asymmetric encryption scheme, on input `key`, return the public key pk.
- On input `encrypt(`m_l`,`m_r`)`, draw a new randomness $r \in \{0,1\}^\eta$ and output $c = \mathcal{E}(m_s, pk, r)$. Record $L := c :: L$.
- On input `decrypt(`c`)`, if $c \in L$ then return `invalid` otherwise return $\mathcal{D}(c, sk)$.

We say that $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ in *IND-CCA2 secure* if for any PPT $\mathcal{A}$ we have

$$\left|\mathbb{P}\left[\mathcal{A}^{\mathcal{O}_l(1^\eta)}(1^\eta) = 1\right] - \mathbb{P}\left[\mathcal{A}^{\mathcal{O}_r(1^\eta)}(1^\eta) = 1\right]\right| = \mathrm{negl}(\eta)$$

where $\mathrm{negl}(\eta)$ is a negligible function in η.

The KDM security defined by [BRS02] is even stronger in the sense that the adversary is able to compute the plaintext depending on the secret keys of the oracle. This notion is not necessary to prove most protocols, however it allows key cycles and thus gives more freedom for designing protocols. We also present it here because the proofs of the axioms are a bit simpler if we assume this notion.

Definition 3.9 (KDM)**.** Let $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme. Let $\mathcal{O}_s(1^\eta)$ (with $s = l$ or $s = r$) be an oracle behaving as follows:

- Generate a vector of key pairs $(pk_1, sk_1), \ldots, (pk_n, sk_n) = \mathcal{K}(1^\eta)^n$, if the encryption scheme is a symmetric encryption scheme, we will take in order to have uniform notation for both cases $pk_i = sk_i$.
- Initialise a list $L :=$ `[]` that memorises all queried encryptions.
- If the encryption scheme is an asymmetric encryption scheme, on input $\texttt{key}_i$, return the public key pk_i.
- On input `encrypt(`p_l`,`p_r`,`i`)`, where p_l and p_r are (encodings of) polynomial time Turing machine:
  - compute $m_s = p_s((pk_1, sk_1), \ldots, (pk_n, sk_n))$ for $s = l, r$
  - if the length of the output of p_s is not constant return `invalid`. If the length of the outputs of p_l and p_r are not equal, return `invalid`.

  - otherwise draw a new randomness $r \in \{0,1\}^\eta$ and output $c = \mathcal{E}(m_s, pk_i, r)$. Record $L := c :: L$
- On input `decrypt(`c`,`i`)`, if $c \in L$ then return `invalid` otherwise return $\mathcal{D}(c, sk_i)$.

We say that $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ in *KDM secure* if for any PPT $\mathcal{A}$ we have

$$\left|\mathbb{P}\left[\mathcal{A}^{\mathcal{O}_l(1^\eta)}(1^\eta) = 1\right] - \mathbb{P}\left[\mathcal{A}^{\mathcal{O}_r(1^\eta)}(1^\eta) = 1\right]\right| = \text{negl}(\eta)$$

where $\text{negl}(\eta)$ is a negligible function in η.

One of the notions of security we do need in order to prove protocols that involve symmetric encryption is the integrity of cyphertexts. We use the most straightforward notion of integrity, namely INT-CTXT (introduced in [BKY01]) which states that the adversary should not be able to forge a new cyphertext from existing ones.

Definition 3.10 (INT-CTXT)**.** Let $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ be a symmetric encryption scheme. Let $\mathcal{O}_k$ be the encryption oracle for key k. The encryption scheme is said *INT-CTXT* if the following quantity is negligible in η

$$\left|\mathbb{P}_{k,r}\left[\mathcal{A}^{\mathcal{O}_k(1^\eta)}(1^\eta) = c \text{ with } \mathcal{D}(c,k) \neq \perp \wedge c \notin S\right]\right|$$

Where $S = \{c_1, c_2, \cdots\}$ is the sequence of answers of the oracle $\mathcal{O}_k(x_i)$

Let us now prove that any of these three notions implies that keys are computationally unpredictable. This is intuitive because if there was a good probability for a key to be guessed, there would be no way for the encryption scheme to be secure.

Proposition 3.11. *Assume that the encryption scheme $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ satisfies any of the aforementioned security properties. If the sort* skey *(resp.* akey *in the asymmetric case) is implemented by the key generation algorithm $\mathcal{K}$ (resp. the first projection of $\mathcal{K}$), the sort* skey *(resp.* akey*) is a computationally unpredictable sort.*

Proof. We only consider the symmetric case, the asymmetric case is treated exactly in the same way. Let us assume by contradiction that there exists $\mathcal{A}$ such that

$$\mathbb{P}_\tau\left[\mathcal{K}(1^\eta) = \mathcal{A}(1^\eta)\right] = f(\eta)$$

where f is non negligible. Now, consider the adversary $\mathcal{B}$ against the IND-CCA game behaving as follows:

- compute $k = \mathcal{A}(\eta)$
- choose two random bitstrings m_1, m_2 of size η
- submit `encrypt(`m_1`,`m_2`)` to $\mathcal{O}$ obtaining c
- if $\mathcal{D}(c,k) = m_1$ return 1, if $\mathcal{D}(c,k) = m_2$ return 0
- otherwise return 0 or 1 randomly

Note that if the key drawn by the oracle is not $k' \neq k$, and if m is a random bitstring, $\mathbb{P}[\mathcal{D}(\mathcal{E}(m, k', r), k) = m]$ is necessarily negligible as otherwise we would be able to reliably decrypt without the key with non-negligible probability which contradict IND-CPA. Let us call $\text{negl}(\eta)$ this probability.

$\mathcal{B}$ answers 1 in the presence of $\mathcal{O}_l$ if one of the following happens:

- $k = k'$
- $k \neq k'$ and either
  - $\mathcal{D}(c,k) = m_1$
  - or $\mathcal{D}(c,k) \neq m_1, m_2$ and 1 is randomly chosen at the last step.

With these considerations, the probability of $\mathcal{B}$ answering 1 in the presence of $\mathcal{O}_l$ is

$$p_l(\eta) = \mathbb{P}[k = k'] + (1 - \mathbb{P}[k = k'])(\underbrace{\text{negl}(\eta)}_{\substack{k \neq k' \\ \mathcal{D}(c,k) = m_1}} + \frac{1}{2} \underbrace{(1 - \text{negl}(\eta))^2}_{\substack{k \neq k' \\ \mathcal{D}(c,k) \neq m_1, m_2}})$$

Similarly, let us compute the probability of $\mathcal{B}$ answering 1 in the presence of $\mathcal{O}_r$

$$p_l(\eta) = (1 - \mathbb{P}[k = k'])(\underbrace{\text{negl}(\eta)}_{\substack{k \neq k' \\ \mathcal{D}(c,k) = m_1}} + \frac{1}{2} \underbrace{(1 - \text{negl}(\eta))^2}_{\substack{k \neq k' \\ \mathcal{D}(c,k) \neq m_1, m_2}})$$

Having computed these two quantities, we get the following contradiction:

$$\left| \mathbb{P}\left[\mathcal{B}^{\mathcal{O}_l(1^\eta)}(1^\eta) = 1\right] - \mathbb{P}\left[\mathcal{B}^{\mathcal{O}_r(1^\eta)}(1^\eta) = 1\right] \right| = p_l - p_r = f(\eta)$$

As IND-CCA2 and KDM are stronger than IND-CPA, we do not need to prove our statement again for these notion. Let us however give the proof in the case in which the scheme is only INT-CTXT secure. The adversary computing $k = \mathcal{A}(\eta)$ and sending $\mathcal{E}(0, k, r)$ to the oracle wins the INT-CTXT game with probability f which contradicts the hypothesis. □

The last notion we need in order to prove our axioms is the notion of "which key concealing encryption"[AR02]. While it may seem unintuitive, such a notion is needed in order to prove our axioms. Such a property is necessary for simplifying some statements. For instance, we wish to simplify $S, \{u\}_k^r \triangleright n$ into $S \triangleright n$ if k is a usable key. Such a simplification is however only sound if $\{u\}_k^r$ does not carry any information on the plaintext, but also no information on the key. "Which key concealing" expresses the latter property. It states roughly that $\{u\}_k^r$ and $\{u\}_{k'}^{r'}$ are indistinguishable.

A more detailed example of that will be given later in this chapter. Note that while the definition we give here is slightly different that the one in [AR02], the two are equivalent.

Definition 3.12 (Which key concealing)**.** Let $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme. Let $\mathcal{O}_s(1^\eta)$ (with $s = l$ or $s = r$) be an oracle behaving as follows:

- pick two keys $k_l = \mathcal{K}(1^\eta)$ and $k_r = \mathcal{K}(1^\eta)$
- on request `encrypt(`m`,`i`)` with $i \in \{l, r\}$ pick a new randomness r and return $\mathcal{E}(m, k_i, r)$
- on request `challenge(`m`)` pick a new randomness r and return $\mathcal{E}(m, k_s, r)$

The encryption scheme $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ is *which key concealing* if for any PPT $\mathcal{A}$

$$\left| \mathbb{P}\left[\mathcal{A}^{\mathcal{O}_l(1^\eta)}(1^\eta) = 1\right] - \mathbb{P}\left[\mathcal{A}^{\mathcal{O}_r(1^\eta)}(1^\eta) = 1\right] \right| = \text{negl}(\eta)$$

where negl is a negligible function.

3.2.2 Syntactic constraints for key usability

Let us first define key-usability constraints for various security notions. Intuitively usable$^N(k, S)$ means that the way the key k is used in S does not corrupt the key k. In other words, a key k is usable in a set of terms if it is possible to build this set of terms using only the oracles available for security property N for encryption and (possibly) decryption with k. Let us give some examples.

Example 3.13. We show some examples of sets of terms in which the key k is usable or unusable for the various cryptographic properties.

- k is CPA-usable in $\{A\}_k^r$ as the encryption could be performed with the encryption oracle
- k is CPA-usable in $\{k\}_{k'}^{r'};\ \{A\}_k^r$ as using a CPA oracle for the first encryption shows that the previous situation is indistinguishable from the situation in which we give $\{0\}_{k'}^{r'};\ \{A\}_k^r$ to the attacker.
- k is not CPA-usable in $\{A\}_k^r;\ \mathrm{sdec}(A, k)$ as the CPA oracle does not allow decryption, however k is CCA-usable.
- k is neither CPA-usable nor CCA-usable in $\{k\}_k^r$ as the oracles do not allow passing their own key as plaintext, it is however KDM-usable.
- k is not usable (for any N) in $\{k\}_{k'}^r; k'$ as it is derivable.

Note that this notion of key usability is quite similar to the notion introduced in [DDMW06] albeit in a somewhat different context. The notion of key usability in [DDMW06] states that a key is usable after execution of a certain key exchange protocol if the cryptographic security (for example IND-CPA) holds even if instead of having the oracle generate a fresh key at the beginning of the game, the key is established by an honest session of the key exchange protocol witnessed by the adversary. In our case, what we try to capture with the key usability constraints is the fact that the cryptographic security still holds even if instead of simply generating the key at the beginning of the cryptographic game, we generate the key and let the adversary observe some messages that may contain the key. In other words it is a kind of passive version of the key usability defined in [DDMW06]. We will not however formally state a security game for this as we are mostly aiming to rewrite terms using this notion.

An important restriction in all the oracles used for defining the cryptographic properties is that the encryption's randomness must be chosen uniformly and independently. This is what we capture in the next definition.

Definition 3.14 (Valid randomness)**.** We say that a term r is a *valid randomness* with respect to a set of terms S, written valid(r, S), if

- r only appears as a third argument of an encryption (symmetric or asymmetric)
- all subterms of the form $\{\_\}_\_^r$ in S are equal.

In words: r is only used as a randomness for encryption and only one encryption with r is performed (although it may be appear several times in S). This constraint does not really restrict the way we are able to use our axioms. Indeed, in practice it is never the case that randomness from an encryption is reused elsewhere. Let us give a small example of this notion.

Example 3.15. The nonce r is a valid randomness in $\{t\}_k^r; \{\{t\}_k^r\}_{k'}^{r'}$, as r is used in two different places but actually it is the whole encryption that is used twice. However r is not a valid randomness in $\{0\}_k^r; \{1\}_k^r$ as r is used to encrypt two different messages.

Plaintext freshness

Let us start by defining a constraint that holds in the cases in which a key is trivially usable. Intuitively, this plaintext freshness constraint ensures that a key is used only in a way that could be mimicked by the oracles provided by the cryptographic properties. We define this notion only for the IND-CPA or IND-CCA case. Indeed in the KDM case it does not really make sense to speak of plaintext freshness (as keys can be used in plaintexts in the KDM case) and it is simpler to directly define key usability.

Definition 3.16 (Plaintext freshness)**.** For k a symmetric key and S a set of terms, we say that k is *CPA plaintext fresh* (resp. *CCA plaintext fresh*) in S, written $\text{pfresh}^{\text{CPA}}(k, S)$ (resp. $\text{pfresh}^{\text{CCA}}(k, S)$) if

- k appears only in encryption key position (resp. encryption and decryption key position)
- for all subterms t of S if $t = \{\_\}_k^r$ then r is a valid randomness with respect to S

For (sk, pk) an asymmetric key pair and S a set of terms, we say that sk is *CPA plaintext fresh* (resp. *CCA plaintext fresh*) in S, written $\text{pfresh}^{\text{CPA}}(sk, S)$ (resp. $\text{pfresh}^{\text{CCA}}(sk, S)$) if sk is fresh (resp. only appears in decryption key position) in S.

Remark 3.17. Note that for the asymmetric encryption case, a secret key is CPA plaintext fresh if and only if it is fresh.

KDM key usability

The simplest key usability notion is the KDM one, albeit stronger than needed for proving most protocols. The intuition behind that notion is that a key is compromised under KDM if and only if it appears outside of a valid encryption with a non compromised key.

Definition 3.18 (KDM usability)**.** The set of *KDM compromised* keys in $S = t_1; \ldots; t_n$ is the smallest set of keys such that: k is *KDM compromised* if and only if there exists p, l such that k appears in t_l at position p and for all p' prefix of p: $C_{p'} \neq \text{enc}$ or $C_{|p'} = \text{enc}(x, k', r)$ with k' compromised or r is not a valid randomness with respect to S. A key k is *KDM usable* if and only if it is not compromised.

There are two reasons for which we started by presenting this notion. First it reflects almost directly the KDM game, which is not so much the case in the other key usability definition. Intuitively it comes from the fact that the KDM game considers a set of keys, and specifies quite precisely their usage (the other games only consider one key at a time) and our key usability notions account for key protecting other keys. The second reason (note that it is a consequence of the first one) is that it is as easy to prove our secrecy axiom with KDM usability as it is to prove it with plaintext freshness.

IND-CPA and IND-CCA key usability

Key usability is a little bit harder to define in the case of IND-CPA and IND-CCA. Aside from the case in which a key is plaintext fresh (and then trivially

usable), a key is usable if it is encrypted by another usable key. The difference with the KDM case is that instead of defining the set of compromised keys as a smallest fixed point, we have to define the set of usable keys as a smallest fixed point. It ensures that a key might not be protected by itself which would not make sense as far as IND-CPA or IND-CCA are concerned.

Let us start by defining how a not necessarily plaintext fresh key might be protected by other keys.

Definition 3.19 (CPA and CCA key protection)**.** We say that k (resp. (sk, pk)) is CPA (or CCA) *protected* by the set of keys (resp. key pairs) K in the set of terms S (written $\text{protect}^{\text{CPA}}(k, K, S)$ or $\text{protect}^{\text{CCA}}(k, K, S)$) if and only if for each t in S we can write

$$t = C[\{m_1\}_{k_1}^{r_1}, \ldots, \{m_l\}_{k_l}^{r_l}]$$

with

- $r_1, \ldots, r_l$ valid randomnesses with respect to S
- for $1 \leq m \leq l$: $k_m \in K$ (resp. $(sk_m, k_m) \in K$)
- CPA case: k only appears in C in encryption key position with a valid randomness with respect to S (resp. does not appear in C)
- CCA case: k only appears in C in encryption key position with a valid randomness with respect to S or decryption key position (resp. only appears in decryption key position) and if $\text{dec}(C_1, C_2)$ is a subterm of C, either C_1 does not contain holes or none of the k_m (resp. sk_m) appear in C_2

Remark 3.20. Note that if k is plaintext fresh in S, then for all K (in particular $K = \emptyset$), k is protected by K in S.

In the IND-CPA case, the condition is pretty straightforward and as IND-CPA oracles do not authorise decryption, we do not have to take care of the possible decryption of an encrypted message. The more complex condition for IND-CCA is designed to avoid the pathological case where $C = \text{dec}(\_, k')$ and $k_1 = k'$ and $m_1 = k$, in which case k is clearly not protected by k' even if it appears under an encryption with k'.

Example 3.21. Let us give some examples of key protection:

- k is protected by k' in $\{k\}_{k'}^{r}; \{A\}_{k}^{r'}$ as both randomnesses are valid and k is safely encrypted under k'
- k is CCA protected in $\{k\}_{k'}^{r}; \text{dec}(A, k); \{A\}_{k}^{r'}$ for the same reasons as above (as using k as a decryption key is now authorised).
- k is not CCA protected in $\text{dec}(\{k\}_{k'}^{r}, k')$ as k' appears as a subterm of a decryption key position.
- k is CCA protected in $\{k\}_{k'}^{r}; \text{dec}(\{A\}_{k'}^{r'}, k')$ as k' only appears as a decryption key in a term that does not need to be encrypted to ensure that k is protected.

Now we can state that a key k is usable, if it is protected by a key that is itself usable even if the key k is insecure, in other terms if it satisfies the following definition.

Definition 3.22 (CPA and CCA key usability)**.** We say that k (resp. (sk, pk)) is CPA (resp. CCA) key usable if it is in the least fixed point of the function $K \mapsto \{k \mid \text{protect}^{N}(k, K, S)\}$ with $N = \text{CPA}$ (resp. $N = \text{CCA}$).

We use this definition of usability in the proofs to progressively replace the encryptions by plaintext-fresh keys with encryptions of zeros (using the security game oracles), thus obtaining a set of terms where the key that were encrypted by plaintext-fresh keys are now plaintext-fresh themselves. This is typically the proof techniques that are used in all soundness results following [AR02]. Note that the condition on decryption keys for the CCA protection can be seen as making sure that we do not submit honestly encrypted messages to the decryption oracle.

3.2.3 Simple secrecy axiom(s)

Having defined the syntactic constraints stating when a key can be safely used for encryption, we can now start defining the axioms reflecting the security properties of the encryption scheme. The first axiom we present here is a secrecy axiom where we assume that keys are plaintext-fresh (or simply usable in the KDM case). In the IND-CPA and IND-CCA case we do not define immediately an axiom where keys are assumed to be usable instead of plaintext-fresh because the soundness proof of such an axiom actually requires to prove this simple secrecy axiom sound. This axiom states that an encryption with a safe key should not leak any information on a secret. It is the axiom originally proposed in [BC12]. The soundness proof of this axiom was flawed in the original paper from Bana and Comon. We fix it here by adding the which-key concealing hypothesis. We also demonstrate here that this hypothesis is actually necessary.

SEC :

$$X; \{x\}_k^r \triangleright n \quad \rightarrow \quad X \parallel C(k, X), \mathrm{fresh}(r, (X; x)) \triangleright n$$

$$\text{with } C = \begin{cases} \mathrm{pfresh}^{\mathrm{CCA}} & \text{in the CCA case} \\ \mathrm{pfresh}^{\mathrm{CPA}} & \text{in the CPA case} \\ \mathrm{usable}^{\mathrm{KDM}} & \text{in the KDM case} \end{cases}$$

Proposition 3.23. *If the encryption scheme is IND-CPA (resp. IND-CCA, resp. KDM) and which key concealing, the corresponding* SEC *axiom is sound.*

Example 3.24. While the *which key concealing* assumption may seem unnecessary, we exhibit a counterexample if this hypothesis is not satisfied. First of all, let us build an IND-CPA encryption scheme that does not verify the which-key concealing hypothesis. Let $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an IND-CPA encryption scheme, m be a bitstring. We let $\mathbf{0}$ be 0^η. We define $(\mathcal{E}', \mathcal{D}', \mathcal{K}')$ as follows:

$$\mathcal{K}'(r.r') = \mathcal{K}(r).\mathcal{K}(r')$$

$$\mathcal{E}'(m, k_1.k_2, b.r) = \begin{cases} k_1.\mathcal{E}(m, k_2, r) \text{ if } m \neq \mathbf{0} \text{ and } b = 1 \\ k_1.\mathcal{E}(m, k_2, r) \text{ if } m = \mathbf{0} \text{ and } b = 0 \\ \mathbf{0}.\mathcal{E}(m, k_2, r) \text{ otherwise} \end{cases}$$

$$\mathcal{D}'(c_1.c_2, k_1.k_2) = \mathcal{D}(c_2, k_2)$$

It is easy to see that the new scheme is still IND-CPA. In addition let us assume that the implementation of the pair is simply the concatenation of messages. We consider the following atom:

$$\{n\}^{r_1}_{\langle k', \pi_1(\{\mathbf{0}\}^r_k)\rangle}, \{1\}^{r'}_k \triangleright n$$

Let us now consider the sampling family $S = \{\tau | [\![r]\!]_\tau = 0.\_,\ [\![r']\!]_\tau = 1.\_\}$. We write $[\![k]\!]_\tau = k_1.k_2$, note that we have both $[\![\{1\}^{r'}_k]\!]_\tau = k_1.c$ for some c and $[\![\{n\}^{r_1}_{\langle k', \pi_1(\{\mathbf{0}\}^r_k)\rangle}]\!]_\tau = d.\mathcal{E}(m, k_1, r)$ for some d. From these observations, we can deduce that $S \models \Box\Diamond\{n\}^{r_1}_{\langle k', \pi_1(\{\mathbf{0}\}^r_k)\rangle}, \{1\}^{r'}_k \triangleright n$ as $\mathcal{D}$ is obviously computable.

Now, let us show why the secrecy axiom does not hold. As k is CPA plaintext fresh, the secrecy axiom would yield

$$S \models \Box\Diamond\{n\}^{r_1}_{\langle k', \pi_1(\{\mathbf{0}\}^r_k)\rangle} \triangleright n$$

which is absurd as on $S' = \{\tau | [\![r]\!]_\tau = 0.\_\}$ the encryption of n is basically an encryption with an honest key and a valid randomness.

One could think of fixing this problem by adding the encryption of zeros to the second part of the secrecy axiom:

$$X; \{x\}^r_k \triangleright n \quad \rightarrow \quad X; \{\mathbf{0}\}^r_k \triangleright n \parallel C(k, X), \mathrm{fresh}(r, X; x)$$

however it is easy to check that the same counterexample holds.

We now proceed to prove a succession of lemmas leading to the proof of proposition 3.23. Each of the following lemmas shows that if a nonce is guessable from X together with an encryption in X then it is guessable from X, with weaker and weaker assumptions on this encryption. For simplicity's sake, when η is clear from the context, we will omit it as an argument of the PPT involved. We start by proving a simple lemma in which the encryption is an encryption of zeros with a fresh key:

Lemma 3.25. *If k and r are fresh in X and $S, \mathcal{A} \models X; \{0\}^r_k \triangleright n$ then there exists a PPT $\mathcal{B}$ and a polynomial p such that:*

- *for all $\tau \in S_\eta$, $\mathcal{B}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ where $\#$ is a fresh delimiter symbol.*
- *$\{\tau \in S_\eta | \mathcal{B}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ with $\forall i. n_i \neq [\![n]\!]_\tau\}$ is negligible.*

Proof. The main idea of the proof is to guess k and r and then simulate $\mathcal{A}$. We will precise the polynomial p later.

On input m, $\mathcal{B}$ draws $k_1, \ldots, k_{p(\eta)}$ and $r_1, \ldots, r_{p(\eta)}$ uniformly at random and outputs $\mathcal{A}(m, \{0\}^{r_1}_{k_1}) \# \ldots \# \mathcal{A}(m, \{0\}^{r_{p(\eta)}}_{k_{p(\eta)}})$

We are left to prove that at least one of the random choices falls in S with overwhelming probability. We will denote by $S^{-k,r}$ the set samplings that differ from a sampling in S only on the interpretation of k and r. First of all, let us note that as S is non negligible, there exists $c \in \mathbb{N}$ and $\alpha > 0$ such that $|S_\eta| \geq \frac{\alpha}{\eta^c}$. As S is non negligible, note that $S^{-k,r}$ is also non negligible because $S \subseteq S^{-k,r}$. Let us also note that

$$\mathbb{P}_\tau(\tau \in S^{-k,r}_\eta \ \wedge \ \tau \in S_\eta) \geq \frac{\alpha}{\eta^c}$$

Therefore, as k, r are fresh in X, we get the following inequality:

$$\mathbb{P}_{\tau,k',r'}(\exists \tau' \in S. [\![k]\!]_{\tau'} = k' \wedge [\![r]\!]_{\tau'} = r' \ \wedge \ [\![X]\!]_{\tau} = [\![X]\!]_{\tau'}) \geq \frac{\alpha}{\eta^c}$$

With this remark we can compute the probability of $\mathcal{B}$ correctly guessing n:

$$\mathbb{P}_{\tau \in S}\big(\mathcal{B}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)} \text{ with } \forall i. n_i \neq [\![n]\!]_\tau\big) \leq (1 - \frac{\alpha}{\eta^c})^{p(\eta)}$$

We now need to prove that $(1 - \frac{\alpha}{\eta^c})^{p(\eta)}$ is a negligible function for a suitable polynomial p.

$$\begin{array}{rcl} \left(1 - \frac{\alpha}{\eta^c}\right)^{p(\eta)} & = & \exp\left(p\left(\eta\right) . \ln\left(1 - \frac{\alpha}{\eta^c}\right)\right) \\ & \sim_{\eta \to \infty} & \exp\left(-p\left(\eta\right) . \frac{\alpha}{\eta^c}\right) \end{array}$$

Choosing $p(x) = x^{c+1}$ is enough make this function negligible. □

Having proven this first lemma, we weaken the assumption on the encryption, letting the key be plaintext-fresh instead of fresh.

Lemma 3.26. *Assume that the encryption scheme is IND-CPA (resp. IND-CCA, resp. KDM) and which key concealing. If k is CPA (resp. CCA, resp. KDM) plaintext fresh and r is fresh in X and $S, \mathcal{A} \models X; \{0\}^r_k \triangleright n$ then there exists a PPT $\mathcal{B}$ and a polynomial p such that:*

- *for all $\tau \in S$, $\mathcal{B}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ where $\#$ is a fresh delimiter symbol.*
- *$\{\tau \in S | \mathcal{B}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ with $\forall i. n_i \neq [\![n]\!]_\tau\}$ is negligible*

Proof. All Turing machines considered here take the security parameter in unary as input. For simplicity's sake we will omit it here as it is clear from context. We will prove this lemma by contradiction using lemma 3.25 and the which key concealing hypothesis. Assume that we have $S, \mathcal{A} \models X; \{0\}^r_k \triangleright n$ under the condition of the lemma. Assume that for all PPT $\mathcal{B}$ we have

$$\{\tau \in S_\eta | \mathcal{B}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)} \text{ with } \forall i. n_i \neq [\![n]\!]_\tau\} \text{ non-negligible} \quad (3.1)$$

We will use the oracle provided by the which-key concealing definition with keys k and k' to build either $X; \{0\}^r_k$ or $X; \{0\}^{r'}_{k'}$. We build X using the left key of the oracle for encryptions with k and submit 0 as the challenge. If the oracle is the left oracle, we have built $X, \{0\}^r_k$, and if we are dealing with the right oracle, we have built $X, \{0\}^{r'}_{k'}$ with k', r' fresh. We can then use lemma 3.25 to build an adversary that distinguishes the two cases.

In more details, let us assume that S is "maximal". More precisely we take $S = \{\tau \mid \mathcal{A}([\![X; \{0\}^r_k]\!]_\tau) = [\![n]\!]_\tau\}$. This is not a loss of generality. Indeed proving the result for such an S is enough to guarantee that it holds for all $S' \subseteq S \cup S_{\text{negl}}$ with S_{negl} of negligible size. Let now $S_0 = \{\tau \mid \mathcal{A}([\![X; \{0\}^{r'}_{k'}]\!]_\tau) = [\![n]\!]_\tau\}$ for some k', r' fresh in X. Lemma 3.25 yields $\mathcal{D}$ such that

$$\{\tau \in S_{0,\eta} | \mathcal{D}([\![X]\!]_\tau) = n_1 \# \ldots \# n_{p(\eta)} \text{ with } \forall i. n_i \neq [\![n]\!]_\tau\} \text{ is negligible}$$

We now consider the following adversary $\mathcal{B}$ against the which key concealing game that behaves as follows:

- Draw all names except k and the randomness involved in k encryptions. Let us call this sampling (together with the oracles interpretation of k, k' and randomnesses) τ.
- Build $[\![X]\!]_\tau$ using the encryption oracle for encryption (and decryption oracle for decryption in the CCA and KDM case) with key k (note that the key freshness or KDM usability hypothesis ensures that this is possible)
- send 0 as the challenge to the which key game, call x the result
- compute $\mathcal{A}([\![X]\!]_\tau, x) = a$ and $\mathcal{D}([\![X]\!]_\tau) = d$
- if $a = \tau(n)$ and none of the components of d are equal to $\tau(n)$ answer `left`
- otherwise answer `right` with probability $\frac{1}{2}$ and `left` with probability $\frac{1}{2}$

Let us now compute the advantage of this adversary. We divide the sampling space as follows:

- $S_d = \{\tau |$ one of the components of $\mathcal{D}([\![X]\!]_\tau)$ is equal to $\tau(n)\}$ of size $\delta_d(\eta)$
- $S_{a00} = \{\tau \notin S_d | \mathcal{A}([\![X, \{0\}^{r'}_{k'}]\!]_\tau) \neq \tau(n)$ and $\mathcal{A}([\![X, \{0\}^{r}_{k}]\!]_\tau) \neq \tau(n)\}$ of size $\delta_{a00}(\eta)$
- $S_{a01} = \{\tau \notin S_d | \mathcal{A}([\![X, \{0\}^{r'}_{k'}]\!]_\tau) \neq \tau(n)$ and $\mathcal{A}([\![X, \{0\}^{r}_{k}]\!]_\tau) = \tau(n)\}$ of size $\delta_{a01}(\eta)$
- $S_{a10} = \{\tau \notin S_d | \mathcal{A}([\![X, \{0\}^{r'}_{k'}]\!]_\tau) = \tau(n)$ and $\mathcal{A}([\![X, \{0\}^{r}_{k}]\!]_\tau) \neq \tau(n)\}$ of size $\delta_{a10}(\eta)$
- $S_{a11} = \{\tau \notin S_d | \mathcal{A}([\![X, \{0\}^{r'}_{k'}]\!]_\tau) = \tau(n) = \mathcal{A}([\![X, \{0\}^{r}_{k}]\!]_\tau)\}$ of size $\delta_{a11}(\eta)$

The definition of S_0 yields $S_{a11}, S_{a10} \subseteq S_0$. With the definition of $\mathcal{D}$, $S_0 \backslash S_d$ is of negligible size. Hence δ_{a11} and δ_{a10} are negligible. By equation 3.1 we have that $\delta_{a01} + \delta_{a11}$ is non negligible, hence δ_{a01} is non negligible.

With these remarks, we can compute the advantage of $\mathcal{B}$ against the which key concealing game. We distinguish the case in which the oracle is the left one or the right one:

- left case: the probability of guessing correctly is

$$\frac{1}{2}.(\delta_d + \delta_{a00} + \delta_{a10}) + \delta_{a01} + \delta_{a11}$$

- right case:

$$\frac{1}{2}.(\delta_d + \delta_{a00} + \delta_{a01})$$

We can then compute the probability α of $\mathcal{B}$ guessing correctly

$$\begin{aligned}
\alpha &= \frac{1}{2}.\delta_d + \frac{1}{2}.\delta_{a00} + \frac{1}{2}.\delta_{a11} + \frac{1}{4}.\delta_{a10} + \frac{3}{4}.\delta_{a01} \\
&= \frac{1}{2}.(\delta_d + \delta_{a00} + \delta_{a01} + \delta_{a10} + \delta_{a11}) + (\frac{1}{4}.\delta_{a01} - \frac{1}{4}.\delta_{a10}) \\
&= \frac{1}{2} + \frac{1}{4}(\delta_{a01} - \delta_{a10})
\end{aligned}$$

As δ_{a01} is non negligible and δ_{a10} is negligible, $\mathcal{B}$ has a non negligible advantage, which contradicts the cryptographic hypothesis. □

We can now proceed to prove the last lemma. Here the restriction on the encryption is the same as the one the of secrecy axiom. Once we have this lemma, it will be enough to show that being able to guess n with good probability from X implies that $X \triangleright n$ to prove proposition 3.23.

Lemma 3.27. *Assume that the encryption scheme is IND-CPA (resp. IND-CCA, resp. KDM) and which key concealing. If k is CPA (resp. CCA, resp. KDM) plaintext fresh and r is fresh in X, x and $S, \mathcal{A} \models X; \{x\}_k^r \triangleright n$ then there exists a PPT $\mathcal{B}$ and a polynomial p such that:*

- *for all $\tau \in S$, $\mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \dots \# n_{p(\eta)}$ where $\#$ is a fresh delimiter symbol.*
- *$\{\tau \in S | \mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \dots \# n_{p(\eta)}$ with $\forall i. n_i \neq \llbracket n \rrbracket_\tau\}$ is negligible*

Proof. This proof is very similar to the proof of lemma 3.26. As previously Assume that $S, \mathcal{A} \models X, \{x\}_k^r \triangleright n$ under the condition of the lemma. Assume that for all PPT $\mathcal{B}$ we have

$$\{\tau \in S | \mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \dots \# n_{p(\eta)} \text{ with } \forall i. n_i \neq \llbracket n \rrbracket_\tau\} \text{ non-negligible} \quad (3.2)$$

Let us assume that S is maximal. Let now S_0 be a maximal sampling such that $S_0, \mathcal{A} \models X; \{0\}_k^r \triangleright n$ for some k', r' fresh. Lemma 3.26 yields $\mathcal{D}$ such that

$$\{\tau \in S_0 | \mathcal{D}(\llbracket X \rrbracket_\tau) = n_1 \# \dots \# n_{p(\eta)} \text{ with } \forall i. n_i \neq \llbracket n \rrbracket_\tau\} \text{ is negligible}$$

Let us call $\mathcal{O}$ the left or right CPA (resp. CCA, resp. KDM) oracle. We now build the following adversary $\mathcal{B}$ against the CPA (resp. CCA, resp. KDM) game that behaves as follows:

- Draw all names expert k and the randomness involved in k encryptions. Let us call this sampling (together with the oracles interpretation of k and randomnesses) τ.
- Build $\llbracket X \rrbracket_\tau$ and $\llbracket x \rrbracket_\tau$ using the encryption oracle (and decryption in the CCA and KDM case) with key k. More precisely, we build $\llbracket u \rrbracket_\tau$ recursively as follows:

$$\llbracket f(u_1, \dots, u_n) \rrbracket_\tau = \begin{cases} \llbracket f \rrbracket(\llbracket u_1 \rrbracket_\tau, \dots, \llbracket u_n \rrbracket_\tau) \text{ if } u \neq \text{enc}(\_, k, \_), \text{dec}(\_, k) \\ \mathcal{O}(\llbracket m \rrbracket_\tau, \llbracket m \rrbracket_\tau) \text{ if } u = \text{enc}(m, k, r) \\ \mathcal{O}(\llbracket m \rrbracket_\tau) \text{ if } u = \text{dec}(m, k) \end{cases}$$

  using a table to compute the interpretation of each term only once. Note that the key plaintext-freshness or KDM usability hypothesis ensures that this is possible, and that we have the correct oracles.
- send $(\llbracket x \rrbracket_\tau, 0)$ to the encryption oracle and call the result e.
- compute $\mathcal{A}(\llbracket X \rrbracket_\tau, e) = a$ and $\mathcal{D}(\llbracket X \rrbracket_\tau) = d$
- if $a = \tau(n)$ and none of the components of d are equal to $\tau(n)$ answer `left`
- otherwise answer `right` with probability $\frac{1}{2}$ or `left` with probability $\frac{1}{2}$

With exactly the same computation as in the previous proof, we obtain a non-negligible advantage for $\mathcal{B}$ against the CPA (resp. CCA, resp. KDM) game which is contradictory. □

We now have all the tools to prove proposition 3.23.

proof of proposition 3.23. Assume that the interpretation of enc is IND-CPA (resp. IND-CCA, resp. KDM). Assume $S \models \Box\Diamond X; \{x\}_k^r \triangleright n$ with k is CPA plaintext fresh (resp. CCA plaintext fresh, resp. KDM usable) in $X; x$ and r fresh in $X; x$.

We need to prove $S \models \Box\Diamond X \triangleright n$. Let $S' \subseteq S$ be a sampling family. The assumption $S \models \Box\Diamond X; \{x\}_k^r \triangleright n$ provides us with a sampling $S'' \subseteq S'$ and $\mathcal{A}$ such that $S'', \mathcal{A} \models X; \{x\}_k^r \triangleright n$. Lemma 3.27 yields $\mathcal{B}$ and p such that

- for all $\tau \in S''$, $\mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ where $\#$ is a fresh delimiter symbol.
- $\{\tau \in S'' | \mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ with $\forall i. n_i \neq \llbracket n \rrbracket_\tau\}$ is negligible

We can now build $\mathcal{B}'$ that on input m

1. computes $p(\eta)$
2. pick $1 \leq i \leq p(\eta)$ uniformly at random (we take a random tape which is not restricted in S'')
3. computes $n_1 \# \ldots \# n_{p(\eta)} = \mathcal{B}(m)$
4. returns n_i

Note that $\mathcal{B}'$ guesses the correct i on a subset of size $\frac{1}{p(\eta)}.(|S''_\eta| - \text{negl}(\eta))$ at least which concludes the proof as it is a non negligible subset of S'. □

We have now proven the soundness of our first secrecy axiom. Let us then give short examples of proofs using this axiom.

Example 3.28. We would like to show that given $\{k'\}_k^r$ and $\{n\}_{k'}^{r'}$ an adversary is not able to compute the nonce n. Let us assume $\{k'\}_k^r; \{n\}_{k'}^{r'} \triangleright n$. Using the secrecy axiom, as k is plaintext fresh and r is a valid randomness, we get $\{n\}_{k'}^{r'} \triangleright n$, and another application of Sec gives us $\emptyset \triangleright n$. Now, as n is trivially fresh in $\emptyset$, the freshness axiom yields a contradiction.

Example 3.29. Now assume we have a key cycle: $\{k'\}_k^r; \{k\}_{k'}^{r'}; \{n\}_{k'}^{r_1} \triangleright n$. This should not be a contradiction if we are only using the CPA or CCA version of Sec and indeed, neither k nor k' are plaintext fresh, therefore we can not apply the secrecy axiom. However if we use the KDM instance, we obtain $\{k'\}_k^r; \{k\}_{k'}^{r'} \triangleright n$ after one application of this axiom contradicting the nonce freshness axiom.

3.2.4 Key usability and full secrecy axioms

The simple secrecy axiom is strong enough to prove most protocols relying only on the secrecy of an encrypted plaintext. The full secrecy axiom we present here is indeed not more powerful than the simple secrecy axiom. It is however more convenient for building proofs. Note that using this axiom instead of the simple secrecy one also yields a non negligible speedup in our tool. We could derive this second secrecy axiom from the previous one, but a direct proof is here slightly more convenient. Indeed it relies mainly on a technical lemma allowing us to rewrite the left hand side of $\triangleright$ atoms that will be heavily reused in the proofs of the integrity axioms.

This Sec_2 axiom is very similar to the Sec axiom, except that it uses key usability instead of plaintext freshness as a constraint. This is a generalisation of the secrecy axiom from [BC12].

$$X, \{x\}_k^r \triangleright n \quad \rightarrow \quad X \triangleright n \parallel \text{usable}^N(k, (X; x)), \text{fresh}(r, (X; x))$$

Where N may be CPA, CCA or KDM depending on the assumption on the encryption scheme.

As usual, we do need to prove that under the suitable cryptographic assumption, this axiom is sound.

Proposition 3.30. *Assuming the encryption scheme is IND-CPA (resp. IND-CPA) and which key concealing,* SEC_2^{CPA} *(resp.* SEC_2^{CCA}*) is sound.*

The direct proof of this proposition relies mainly on lemma 3.27, that allows us to explicitly build a PPT computing n from X given a PPT computing n from $X; \{x\}_k^r$. The main difficulty is to get rid of the occurrences of k inside encryptions. This is exactly the purpose of the following lemma. Note that this lemma allows us to simplify the left hand side of $\triangleright$ atoms in order to prove the integrity axioms in the next subsection. It is worth remarking that one way of looking at this lemma is that if a key is usable, then for any purpose it is as good as a plaintext fresh key, which is quite similar to the idea of key usability from [DDMW06].

Lemma 3.31. *Assume the encryption scheme is IND-CPA (resp. IND-CCA). Let X be a set of terms, let K be a set of CPA (resp. CCA) usable keys in X. We have*

$$\mathbb{P}_\tau\big[\mathcal{A}([\![X]\!]_\tau) = 0\big] \quad = \quad \mathbb{P}_\tau\big[\mathcal{A}([\![\widetilde{X}]\!]_\tau) = 0\big] + \text{negl}$$

When $\widetilde{X}$ is obtained from X by rewriting any encryption with a key in K to an encryption of zeros. Note that handles are computed by a PPT from their dependencies. Therefore it makes sense to replace encryptions in these dependencies.

Proof. This proof is very similar to the proof in [AR02], we however consider an arbitrary signature in addition to the encryption, and stay in the computational model.

We reason by induction on the maximal length of the derivation of $\text{usable}(k, X)$ for the keys in K. We write $K = K_0 \sqcup \cdots \sqcup K_s$ where K_i is the set of keys k for which i is the minimal length of the derivation of $\text{usable}(k, X)$. We assume $K_s \neq \emptyset$. Let us write $K_s = \{k_1, \ldots, k_l\}$. Let $\mathcal{O}$ be an IND-CPA (resp. IND-CCA) oracles. Let $\mathcal{A}$ be a PPT. We write

$$\epsilon = \mathbb{P}_\tau\big[\mathcal{A}([\![X]\!]_\tau) = 0\big] - \mathbb{P}_\tau\big[\mathcal{A}([\![\widetilde{X}]\!]_\tau) = 0\big]$$

Base case: if $s = 0$ for all keys in K, it means that all keys in K are CPA (resp. CCA) plaintext fresh. We use a classical hybrid argument. We define X_m for $0 \leq m \leq l$ as X rewritten with the rules $\text{enc}(x, k_i, y) \to \text{enc}(0, k_i, y)$ for $1 \leq i \leq m$. Note that $X_0 = X$ and $X_l = \widetilde{X}$. Let us now prove that for $0 < m \leq l$

$$\mathbb{P}_\tau\big[\mathcal{A}([\![X_{m-1}]\!]_\tau) = 0\big] \quad = \quad \mathbb{P}_\tau\big[\mathcal{A}([\![X_m]\!]_\tau) = 0\big] + \text{negl}$$

Let us call $\epsilon_m = \mathbb{P}_\tau[\mathcal{A}([\![X_{m-1}]\!]_\tau) = 0] - \mathbb{P}_\tau[\mathcal{A}([\![X_m]\!]_\tau) = 0]$

Let us build the following adversary $\mathcal{B}$ against the IND-CPA (resp. IND-CCA) game. $\mathcal{B}$ behaves as follows:

- recursively compute the interpretation $r(X)$ of X_{m-1} or X_m using $\mathcal{O}$:
  - draw the interpretation τ of all constants apart from k_m and the nonces that appear as randomness of k_m encryption.

– if $X = \{u_1, \ldots, u_k\}$ compute $r(X) = \{r(u_1), \ldots, r(u_k)\}$ with $r(u)$ defined by

$$r(u) = \begin{cases} \tau(u) \text{ if } u \text{ is a constant} \\ [\![g]\!](r(v_1), \ldots, r(v_l)) \text{ if } u = g(v_1, \ldots, v_l) \text{ with } g \neq \text{enc}, \text{dec} \\ \mathcal{E}(r(v_1), r(v_2), r(v_3)) \text{ if } u = \text{enc}(v_1, v_2, v_3) \text{ with } v_2 \notin \{k_1, \ldots, k_m\} \\ \mathcal{E}(0, r(v_2), r(v_3)) \text{ if } u = \text{enc}(v_1, v_2, v_3) \text{ with } v_2 \in \{k_1, \ldots, k_{m-1}\} \\ \mathcal{O}(r(v_1), 0) \text{ if } u = \text{enc}(v_1, k_m, v_3) \\ \mathcal{D}(r(v_1), r(v_2)) \text{ if } u = \text{dec}(v_1, v_2) \text{ with } v_2 \neq k_m \\ \mathcal{O}(r(v_1)) \text{ if } u = \text{dec}(v_1, k_m) \end{cases}$$

It is easy to check that CPA (resp. CCA) plaintext freshness implies that k_m is not used outside of the oracle calls, and that in the CPA case the decryption oracle is never used. Additionally, the randomnesses used in k_m encryptions are all used only once.

- compute $\mathcal{A}(r(X)) = a$
- if $a = \tau(n)$ answer `left` otherwise answer `right`

It is clear from the previous remarks that for τ, if the CPA (resp. CCA) oracle is the left oracle then $r(X) = [\![X_{m-1}]\!]_\tau$ and if it is the right oracle, then $r(X) = [\![X_m]\!]_\tau$. We now compute the advantage of $\mathcal{B}$ against the IND-CPA (resp. IND-CCA) game:

$$\begin{aligned} \text{Adv}(\mathcal{B}) &= \mathbb{P}_{\tau,b}[\mathcal{A}(r(X)) = 0, b = \texttt{left}] + \mathbb{P}_{\tau,b}[\mathcal{A}(r(X)) \neq 0, b = \texttt{right}] \\ &= \frac{1}{2}.\big(\mathbb{P}_{\tau,b}[\mathcal{A}([\![X_{m-1}]\!]_\tau) = 0] + \mathbb{P}_{\tau,b}[\mathcal{A}([\![X_m]\!]_\tau) \neq 0]\big) \\ &= \frac{1}{2} + \frac{1}{2}\epsilon_m \end{aligned}$$

The cryptographic hypothesis yields the fact that ϵ_m is negligible. It is now enough to note that $|\epsilon| \leq |\epsilon_1| + \cdots + |\epsilon_s|$ to get the fact that ϵ is negligible.

Induction step: Let us assume that for all K such that the usability of any key in k can be derived in at most $s-1$ step, the lemma is true. Consider $K' = K_0 \sqcup \ldots K_{s-1}$. We call X' the set X in which all terms are rewritten with the rules $\text{enc}(x, k, z) \to \text{enc}(0, k, z)$ for all k in K'. From the induction hypothesis we get that

$$\epsilon_1 = \mathbb{P}_\tau[\mathcal{A}([\![X]\!]_\tau) = 0] - \mathbb{P}_\tau[\mathcal{A}([\![X']\!]_\tau) = 0]$$

is negligible. Let us define

$$\epsilon_2 = \mathbb{P}_\tau[\mathcal{A}([\![X']\!]_\tau) = 0] - \mathbb{P}_\tau[\mathcal{A}([\![\widetilde{X}]\!]_\tau) = 0]$$

Let us prove that all keys in K_s are CPA (resp. CCA) plaintext fresh. These keys may not be plaintext fresh in X (otherwise they would be in K_0). Let k be a key in K, let L be a set such that (k, L, X) such that all keys in L can be deemed usable in less than $s-1$ steps. Such a set exists as k can be deemed usable in less than s steps. By maximality of K, we have that $L \subseteq K'$. By definition of key protection, k is plaintext fresh in X'. We now apply the base case to conclude that ϵ_2 is negligible. The induction variable s does not depend on the security parameter. Therefore the union of s negligible sets is negligible. As $|\epsilon| \leq |\epsilon_1| + |\epsilon_2|$, we have that ϵ is negligible which concludes the proof. □

As in the proof of our simple secrecy axiom, we prove a slightly stronger lemma, which will lead to the proof of proposition 3.30.

Lemma 3.32. *Assume that the encryption scheme is IND-CPA (resp. IND-CCA, resp. KDM) and which key concealing. If k is CPA (resp. CCA) usable fresh and r is fresh in X, x and $S, \mathcal{A} \models X, \{x\}_k^r \triangleright n$ then there exists a PPT $\mathcal{B}$ and a polynomial p such that:*

- *for all $\tau \in S$, $\mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ where is a fresh delimiter symbol.*
- *$\{\tau \in S | \mathcal{B}(\llbracket X \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ with $\forall i. n_i \neq \llbracket n \rrbracket_\tau\}$ is negligible*

Proof. Let us choose $X, x, k, r, \mathcal{A}, S$ under the conditions of the lemma, assume S maximal. Let K be the maximal set of usable keys in X. As previously, let us write $\widetilde{X}$, the set obtained from X by rewriting encryptions with usable keys to encryptions of zeros. Lemma 3.31 provides us with S_0 such that $|S_0| = |S|$ (in particular non negligible) such that $S_0, \mathcal{A} \models \widetilde{X}, \{0\}_k^r \triangleright n$. Lemma 3.26 yields $\mathcal{D}$ and p such that

- for all $\tau \in S_0$, $\mathcal{D}(\llbracket \widetilde{X} \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ where is a fresh delimiter symbol.
- $\{\tau \in S_0 | \mathcal{D}(\llbracket \widetilde{X} \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)}$ with $\forall i. n_i \neq \llbracket n \rrbracket_\tau\}$ is negligible

To conclude the proof, we now need to prove that

$$\{\tau \in S | \mathcal{D}(\llbracket X \rrbracket_\tau) = n_1 \# \ldots \# n_{p(\eta)} \text{ with } \forall i. n_i \neq \llbracket n \rrbracket_\tau\} \text{ is negligible}$$

Consider the following PPT $\mathcal{B}$ which on input x, y, z:

- computes $\mathcal{A}(x, y) = a$ and $\mathcal{D}(x) = d_1 \# \ldots \# d_{p(\eta)}$
- answers 0 if $a = z$ and $\forall i. d_i \neq z$, 1 otherwise

Lemma 3.31 yields:

$$\mathbb{P}[\mathcal{B}(X, \{x\}_k^r, n) = 0] = \mathbb{P}[\mathcal{B}(\widetilde{X}, \{0\}_k^r, n) = 0] + \text{negl}$$

As $\mathbb{P}[\mathcal{B}(\widetilde{X}, \{0\}_k^r, n) = 0]$ is negligible, so is $\mathbb{P}[\mathcal{B}(X, \{x\}_k^r, n) = 0]$ which concludes the proof. □

We complete the proof of proposition 3.30 with the exact same proof as the proof of proposition 3.23, using the previous lemma instead of lemma 3.27.

We now come back to example 3.28 to demonstrate that this new secrecy axiom is indeed more efficient for deriving contradictions.

Example 3.33. Consider the atom $\{k'\}_k^r; \{n\}_{k'}^{r'} \triangleright n$. Using the simple secrecy axiom, we need two steps to derive a contradiction. However using this stronger axiom, we get that k' is usable and therefore derive (in one step) $\{k'\}_k^r \triangleright n$ which is a contradiction.

3.2.5 Integrity and non-malleability axiom(s)

While the secrecy axioms are sometimes strong enough to prove simple protocols, most protocols assume that the adversary may not interfere with an honest encryption. This is the point of security properties like IND-CCA which intuitively implies that the adversary is not be able to produce the encryption of a plaintext that is meaningfully related to a plaintext of an honestly generated encryption. The integrity properties for symmetric encryptions are

even stronger, stating that the adversary should not be able to produce valid cyphertexts at all. In what follows we give axioms that reflect these intuitions.

Let us here give a short example of how the malleability of an encryption scheme might be used to break a simple one-way authentication protocol that would be otherwise secure.

Example 3.34. Let us consider the following one-way authentication protocol:

$$\begin{array}{rcrl} A & \to & B: & \{n, A\}_{pk_B} \\ B & \to & A: & \{n\}_{pk_A} \end{array}$$

At the end of the protocol, we would like n to be secure. Let us consider the trace in which B thinks its message originated from the adversary. If the encryption scheme is malleable, there is an attack on this trace:

$$\begin{array}{rcrl} A & \to & I(B): & \{n, A\}_{pk_B} \\ I & \to & B: & \{n, I\}_{pk_B} \\ B & \to & I: & \{n\}_{pk_I} \\ I(B) & \to & A: & \{n\}_{pk_A} \end{array}$$

Where the second message is obtained using the malleability of the encryption scheme. Therefore we can observe that this protocol could not be proven secure with only the secrecy axiom. Now remark that if the encryption scheme is IND-CCA secure, and the key pk_B is honest, the second message in the attack can not depend on n, therefore this trace can not be completed. Formally this attack trace (up to the point where the attacker is able to compute n) is represented by the following set of atoms:

- $\{n, A\}^{r_1}_{pk_B} \triangleright h_1$ for the first attacker's message
- $\pi_2(\text{adec}(h_1, sk_B)) = I$ for the condition tested by B
- $\{n, A\}_{pk_B};\ \{\pi_1(\text{adec}(h_1, sk_B))\}^{r_2}_{pk_I} \triangleright n$ for the derivability of n.

Any reasonable definition of a non-malleability axiom should ensure that these formulae, together with the axioms, are unsatisfiable.

Non-malleability

We define the following non malleability axiom scheme (NM):

$$\boxed{X \triangleright y \quad X, \text{dec}(y, sk) \triangleright n \quad \to \quad \bigvee_{\{x\}^r_{pk} \in \text{st}(X)} y = \{x\}^r_{pk} \parallel C}$$

with C the conjunction of the following constraints:

- $\text{usable}^{\text{CCA}}(sk, X)$
- n only appears under encryption by usable keys of X

Formally the conjunction as conclusion of the axiom is expressed as a constraint. The formula $\bigvee_{\{x\}^r_{pk} \in \text{st}(X)} y = \{x\}^r_{pk}$ represents the formula $\bigvee_{i=1}^{p} y = \{x_i\}^{r_i}_{pk}$ constrained by "$\{\{x_i\}^{r_i}_{pk} | i \leq p\}$ is the set of subterms of X that are encryptions with pk". We abbreviate for simplicity reasons.

The original non-malleability axiom from [BAS12] has the form

$$X \triangleright y \quad X, \text{dec}(y, sk) \triangleright n \quad \to \quad X \triangleright n \ \vee \bigvee_{\{x\}^r_{pk} \in \text{st}(X)} y = \{x\}^r_{pk} \parallel C$$

where the constraint C does not constrain the place where n might appear in X. The computational soundness of this axiom assumes that the signature of the model only contains one type of encryption and pairing. This restriction would not allow to prove protocols that use more primitives than just asymmetric encryption. This is the main reason for which we choose to add an additional constraint on n instead.

Moreover the original axiom assumes plaintext-freshness of the key. We weaken this assumption to key-usability, in order to allow exchange sending private keys around.

Example 3.35. Let us observe that our axiom allows us to prove that the simple one way authentication presented in example 3.34 is secure. By simplifying the second atom $\{n, A\}^{r_1}_{pk_B};\ \{\pi_1(\mathrm{adec}(h_1, sk_B))\}^{r_2}_{pk_I} \triangleright n$, using the functionality axiom, we get $\{n, A\}^{r_1}_{pk_B};\ \mathrm{adec}(h_1, sk_B);\ pk_I \triangleright n$. Using the non-malleability axiom and the atom $\{n, A\}^{r_1}_{pk_B} \triangleright h_1$ we get $h_1 = \{n, A\}^{r_1}_{pk_B}$ and therefore $A = I$ with the condition tested by B. This is a contradiction as A is assumed honest.

Proposition 3.36. *If the encryption scheme is IND-CCA, then the non-malleability axiom is sound.*

Proof. Let us consider an instance of the non-malleability axiom

$$X \triangleright y \quad X, \mathrm{dec}(y, sk) \triangleright n \quad \rightarrow \bigvee_{\{x\}^r_{pk} \in \mathrm{st}(X)} y = \{x\}^r_{pk}$$

where X is a ground extended term and y is a ground term. Assume sk is usable in X and all instances of n in X appear under encryptions by usable keys. Assume $S, \mathcal{A} \models X \triangleright y$ and $S, \mathcal{B} \models X, \mathrm{dec}(y, sk) \triangleright n$ and $S \models \bigwedge_{\{x\}^r_{pk} \in st(X)} y \neq \{x\}^r_{pk}$. Let f be a fresh function symbol, with $[\![f]\!] = \mathcal{A}$. The previous statement is equivalent to $S, \mathcal{B} \models X, \mathrm{dec}(f(X), sk) \triangleright n$ (in other terms we can drop the first premise of the axiom). Assume that the minimal derivation of $\mathrm{usable}^{\mathrm{CCA}}(sk, X)$ ends by using $\mathrm{protect}^{\mathrm{CCA}}(sk, K, X)$. Let us write $X \downarrow_K$ the set X in which all encryptions by keys in K are replaced by encryptions on zeros. We know that $sk \notin K$ therefore, lemma 3.31 yields S_0 such that $|S_0| = |S|$ and $S_0, \mathcal{B} \models X \downarrow_K, \mathrm{dec}(f(X \downarrow_K), sk) \triangleright n$ and (as it is also testable by a PPT) $S_0 \models \bigwedge_{\{x\}^r_{pk} \in st(X\downarrow_K)} y \neq \{x\}^r_{pk}$.

Now we need to prove that there exists S_1 such $|S_1| = |S_0|$ and $S_1, \mathcal{B} \models X \downarrow_{K,sk}, \mathrm{dec}(f(X \downarrow_{K,sk}), sk) \triangleright n$. Let us build the following adversary $\mathcal{D}$ against the IND-CCA game behaving as follows:

1. compute the interpretation $r(X)$ of $X \downarrow_K$ or $X \downarrow_{K,sk}$ submitting $(m, 0)$ to the IND-CCA oracle to compute any pk encryption as in lemma 3.31 and drawing all other constants according to τ.
2. submit $\mathcal{A}(r(X))$ to the IND-CCA oracle, if it returns an error, return `right`, otherwise if it returns m check whether $\mathcal{B}(r(X), m) = \tau(n)$ if so, return `left`, otherwise return `right`.

As in lemma 3.31, we conclude from the IND-CCA security of the encryption that

$$\mathbb{P}_\tau(\mathcal{B}([\![X \downarrow_K, \mathrm{dec}(f(X \downarrow_K))]\!]_\tau = \tau(n))) \approx \mathbb{P}_\tau(\mathcal{B}([\![X \downarrow_{K,sk}, \mathrm{dec}(f(X \downarrow_{K,sk}))]\!]_\tau = \tau(n)))$$

Now, let K_M the maximal set of usable keys in X, lemma 3.31 yields S_2 such that $|S_2| = |S_1|$ and $S_2, \mathcal{B} \models X\downarrow_{K_M}, \mathrm{dec}(f(X\downarrow_{K_M}), sk) \triangleright n$. Now as the FRESH axiom is sound and n is fresh in $X\downarrow_{K_M}$, S_2 is negligible, it follows that S is negligible which concludes the proof. □

Unforgeability

Remark that while this non-malleability axiom could be defined for symmetric encryption as well (and easily proven sound), in practice when speaking of integrity of symmetric encryption, one thinks of unforgeability, which is much stronger than non-malleability as it ensures that the cyphertext was honestly created. We therefore define here an axiom for this unforgeability notion.

Defining this unforgeability axiom requires some more definitions and assumptions on the computational interpretations. We assume that the decryption function returns a public failure constant $\perp$ when failing. We define a well-formedness predicate wf. The computational interpretation of wf is $\mathrm{wf}(x) = (x \neq \perp)$. We do not give axioms for wf. The protocol conditions should include checking that the terms used during its run are well-formed.

Let us now define our integrity axiom scheme INT.

$$X \triangleright y \quad \mathrm{wf}(\mathrm{sdec}(y,k)) \quad \rightarrow \bigvee_{\{z\}_k^r \in \mathrm{st}(X)} y = \{z\}_k^r \parallel C$$

Where as usual C is either $\mathrm{usable}^{\mathrm{CCA}}(k,X)$ or $\mathrm{usable}^{\mathrm{KDM}}(k,X)$. Note that we do not need to consider the CPA case as joint IND-CPA and INT-CTXT implies IND-CCA (see [BN08]).

Note that [BHO13] simultaneously proposed another computationally sound axiom for integrity. However the definition of their an axiom requires dealing with several versions of the $\triangleright$ predicate, for deducibility with respect to various adversaries. We favoured here the simplicity of the model at the cost of having slightly more complicated constraints. As mentioned earlier, this choice allows us to obtain a relatively simple decision procedure.

Let us give a small example of the usefulness of such an axiom.

Example 3.37. Consider the following simple protocol where we wish to guarantee the secrecy of a nonce n. In this protocol A sends a fresh key to B which is then used to encrypt a secret.

$$\begin{array}{rcll} A & \rightarrow & B: & \{k\}_{k_{AB}}^{r_1} \\ B & \rightarrow & A: & \{n\}_k^r \end{array}$$

It is quite easy to see that this protocol may not be proven secure with our non-malleability axiom. Indeed, NM does not guarantee any form of authenticity of an encryption therefore it may not forbid the following attack where the adversary simply forges an encryption.

$$\begin{array}{rcll} I(A) & \rightarrow & B: & \{k_I\}_{k_{AB}}^{r_1} \\ B & \rightarrow & I(A): & \{n\}_{k_I}^r \end{array}$$

However, intuitively this protocol should be secure as long as the encryption scheme ensures unforgeability as the only key that B may use to encrypt is then

k. Let us show that the unforgeability axiom is strong enough to prove this protocol secure. Let us consider the trace where A plays then B (no other trace is really interesting), the corresponding set of formulae is:

1. $\{k\}^{r_1}_{k_{AB}} \triangleright h$ for the first message
2. $\mathrm{wf}(\mathrm{sdec}(h, k_{AB}))$ for B checking that the message decrypts correctly
3. $\{k\}^{r_1}_{k_{AB}};\ \{n\}^{r}_{\mathrm{sdec}(h,k_{AB})} \triangleright n$ for the security property

From atoms 1 and 2 and our unforgeability axiom, we get $h = \{k\}^{r_1}_{k_{AB}}$. Therefore, atom 3 becomes $\{k\}^{r_1}_{k_{AB}};\ \{n\}^{r}_{k} \triangleright n$ and simply applying secrecy and freshness we get a contradiction.

Considering the previous example, one might wonder whether plaintext freshness might be substituted for usability. Consider a modification of the previous protocol where k_{AB} is obtained by executing a key exchange protocol. Now the first atom becomes $X_{k_{AB}};\ \{k\}^{r_1}_{k_{AB}} \triangleright h$ where $X_{k_{AB}}$ is the frame corresponding to the execution of the key exchange protocol. Typically k_{AB} will be usable but not plaintext fresh in $X_{k_{AB}}$. As the right hand side of this atom is not a nonce, there is no way to use secrecy to remove encrypted occurrences of k_{AB}, therefore proving this protocol secure requires the constraint for Int to be key usability.

Of course, having shown the usefulness of this axiom is not enough, we need it to be computationally sound as well. This is the point of the next proposition.

Proposition 3.38. *Assume that the decryption function returns a public failure constant when the encryption is not well formed and the interpretation of* wf *returns true on all bitstrings except this failure constant. Then, provided that the encryption scheme is jointly IND-CCA (resp. KDM) and INT-CTXT, the CCA (resp. KDM) version of* Int *is sound.*

Proof. Let us prove the proposition by contradiction. Assume $S, \mathcal{A}$ are such that:

- $S, \mathcal{A} \models X \triangleright y$
- $S \models \mathrm{wf}(\mathrm{dec}(y, k))$
- $S \models \forall z, r.(\mathrm{enc}(z, k, r) \in \mathrm{st}(X) \rightarrow y \neq \mathrm{enc}(z, k, r))$
- K a set of usable keys such that (K, k, X) and $k \notin K$.

Let L_p be the list of positions in $X\downarrow_K$ that are encryptions with K, let $L = (X_{|_p})_{p \in L_p}$. We only consider the encryptions that are present in $X\downarrow_K$ to ensure that we will be able to build those encryptions with oracles, for example we do not wish to consider $\{k\}^r_k$ if it only appears protected by a fresh key.

Let $\mathcal{D}$ by the PPT behaving as follows on input x_X, x_L, x_k:

1. computes $y = \mathcal{A}(x_X)$
2. answers 1 if $y \in x_L$
3. answers 0 if $\mathrm{wf}(\mathrm{sdec}(y, x_k))$, 1 otherwise. The computational semantics of wf ensures that $\mathcal{D}$ answers 0 at this step if and only if $\mathrm{sdec}(y, x_k) \neq \bot$.

Note that on S, $\mathcal{D}([\![X]\!], [\![L]\!], [\![k]\!])$ returns 0.

By lemma 3.31,

$$\mathbb{P}_\tau(\mathcal{D}([\![X]\!]_\tau, [\![L]\!]_\tau, [\![k]\!]_\tau) = 0) \approx \mathbb{P}_\tau(\mathcal{D}([\![X\downarrow_K]\!]_\tau, [\![L\downarrow_K]\!]_\tau, [\![k]\!]_\tau) = 0)$$

Now, let us remark that as k is protected by K plaintext fresh in $X\downarrow_K$, this means that as usual we can build $X\downarrow_K$ using only the relevant oracles for the

k encryption. Let us build the following adversary $\mathcal{D}'$ against the INT-CTXT game:

1. draws the interpretation of all constants except k and the randomnesses in k encryptions
2. computes the interpretation x_X of $X{\downarrow_K}$ using the encryption/decryption oracle for encrypting/decrypting with k
3. computes $y = \mathcal{A}(x_X)$
4. submits y as the challenge to the INT-CTXT oracle

Let us remark that $\mathcal{D}'$ wins the INT-CTXT game having drawn the constants according to τ if and only if $\mathcal{D}$ outputs 0 on $[\![X{\downarrow_K}]\!]_\tau, [\![L{\downarrow_K}]\!]_\tau, \tau(k)$. It follows that $\mathbb{P}_\tau(\mathcal{D}([\![X{\downarrow_K}]\!]_\tau, [\![L{\downarrow_K}]\!]_\tau, [\![k]\!]_\tau) = 0)$ is negligible, which in turn implies that S is negligible. □

3.3 Conclusion

We have defined here axioms for the usual properties of encryption schemes. As it will be discussed in chapter 5, these axioms are sufficient to prove the protocols in the Spore[SPO14] library, that involve encryption only. The main contribution is the design of syntactic constraints for key usability, that are both strong enough for all reasonable use cases and easily computable. We also formulated new axioms for unforgeability of encryption and modified the non-malleability axiom from [BC12]. Finally we filled a gap in the proof of the secrecy axioms (and further extended them with our key-usability notion) in the symmetric case, showing that the notion of which-key concealing encryption was necessary.

We are confident that the proof techniques used to prove the cryptographic axioms (in particular lemma 3.31) could be widely reused in order to prove sound other axioms for similar properties, typically unforgeability of signatures, or secrecy preserving of zero knowledge proofs. However building a larger library of axioms is left as future work.

Moreover, these axioms, under their current form are well suited for automated consistency checks as will be precised in chapter 5. They do not fit exactly in the class described in chapter 4. The main difference is the fact that the key-usability constraints are not monotone constraints and that the integrity axioms allow a branching on the equational theory. They are however close enough to let us reuse the main intuitions of our decision procedure in the tool.

Chapter 4

Decision procedure

In this chapter, we prove tractability of fragments of First Order Logic. Following the approach of D. Mc Allester [McA93], D. Basin and H. Ganzinger [BG01] show that, if a set of Horn clauses is saturated, with respect to a well suited ordering and a well suited notion of redundancy, then the associated inference system is tractable. They also consider a variadic predicate, however loosing tractability. The main restriction in order to have a **PTIME** decision procedure using this technique is on the ordering with respect to which the clauses have to be saturated: given a ground term t, there should be only polynomially many terms smaller than t. (The subterm ordering, is an example. The term embedding does not satisfy this property).

However, the Horn clauses derived from security assumptions are beyond the scope of these results for several reasons that we describe below.

- The deducibility predicate $\triangleright$ can be seen as a variadic predicate symbol, whose arguments (except the last one) are unordered. This is a problem, since Basin and Ganzinger method yields an NP decision procedure with such a predicate: even if A is saturated (modulo the set axioms for the left part of the $\triangleright$ predicate), when we use A to reduce a ground atom $S \triangleright t$, potentially all subsets of S will be considered (see Section 4.2 for an example).
- Axioms (i.e. Horn clauses) are constrained. A priori, this is not an obstacle to the Basin and Ganzinger procedure, as the constraints can be checked on each superposition between an axiom and a ground clause. However, the very notion of saturation of a set of constrained clauses is an issue (as reported for instance in [NR01] for basic strategies or [CT97] for order constraints). In short: we cannot assume our set of axioms to be saturated.
- Clauses contain an equality predicate. This is not too tricky, since we may assume that A does not contain any equality. Hence equalities appear only as ground literals. We have to consider our clauses modulo a ground equational theory.

4.1 Overview of the results and proofs

We split our results in three parts: we extend stepwise the tractable fragment of First Order Logic. Each extension may use the previous one as a tractable oracle.

Including a variadic predicate. We consider sets of ground Horn clauses with equality, whose atomic formulae may (also) be $S \triangleright t$ where S is a finite set of (ground) terms and t is a ground term, together with a saturated set of clauses $\mathcal{A}$ with no deducibility predicate and the following set of clauses A_0:

$$A_0 = \begin{cases} X \triangleright x & \rightarrow & X; y \triangleright x & \text{weakening} \\ X \triangleright x, \quad Y; x \triangleright y & \rightarrow & X; Y \triangleright y & \text{transitivity} \\ & \rightarrow & x \triangleright x & \\ X_1 \triangleright x_1, \ldots, X_n \triangleright x_n & \rightarrow & X_1; \ldots; X_n \triangleright f(x_1, \ldots, x_n) & f \text{ function symbol} \end{cases}$$

We prove first that satisfiability of such a set of clauses is in PTIME, therefore extending Basin and Ganzinger result, in particular with the deducibility predicate.

The main idea then is to use another layer of ground Horn clauses entailment problem: given $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n, S \triangleright t$, whether $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n$ entails $S \triangleright t$ can be solved in PTIME. This is done by transforming literals $S \triangleright t$ into clauses $S \rightarrow t$. Since the resulting clauses do not contain $\triangleright$ anymore, this can be used as an oracle in a (modified) ground Horn clauses entailment problem. This tractability result is proved stepwise in section 4.2.

Adding axioms on the deducibility predicate. The previous result is not sufficient for our purpose as, for instance, simple axioms such as secrecy (provided in chapter 3) cannot be expressed in the considered fragment.

We therefore extend the previous results, adding formulae of the form

$$S \triangleright x, \quad S; u(x) \triangleright t(y) \quad \rightarrow \quad S \triangleright t(y)$$

$$S; u(x) \triangleright v(y) \quad \rightarrow \quad S \triangleright v(y)$$

These formulae are relevant for our application. Indeed, the secrecy axiom described in chapter 3 is an axiom of the second form. The axioms of the first form are useful to express for example some integrity properties. If we do not take into account the branching on possible equations on right hand side of the axiom, the original non-malleability axiom from [BC12] is of the second form. Likewise, as will be seen in chapter 5, we will use a consequence of our integrity axiom (still ignoring the branching on equations) that has the following form:

$$\forall X, x, y. \quad X \triangleright x \quad X; \mathrm{sdec}(x, k) \triangleright n \quad \rightarrow \quad X \triangleright n \qquad \| \quad P(x)$$

We do not think that ignoring the branching on equations is a problem as the complexity arising from the exploration of all possible equations between decrypted terms and honest encryptions seems to be manageable. This is experimentally confirmed as will be explained in more details in chapter 5.

We show again in this case that the satisfiability is in PTIME. The first idea consists in seeing these clauses as new inference rules. For instance, the first

axiom above can be seen as a generalised cut (it is a cut when $u(x) = x$). As before, we first consider the entailment problem for deduction atomic formulae, which in turn can be seen as an entailment problem for Horn clauses. This can also be easily reduced to the problem of deducing the empty clause.

We design a strategy, which is complete for this extended deduction system and for which the proof search is in PTIME. Let us explain how it works. With the usual cut rule (and not the extended one above), whether the empty clause can be derived, can be decided in PTIME using a unit strategy. This is not the case with an extended cut rule. However, introducing some new rules and additional syntactic constructions, we design a proof system, whose expressive power is the same as the original proof system, and for which the unit strategy is complete, yielding a PTIME decision procedure. In other words, our strategy, that cannot be explained as a local strategy of application, can be reduced to a unit strategy, thanks to some memorisation. This extension, including axioms on the deducibility predicate, is proved in section 4.3.

Adding constraints. Our application case requires to consider constraints, typically expressing that some term does not occur in the left side of a deduction relation. Such constraints have good stability properties: if they are satisfied by two sets of literals, then they are satisfied by their union and, if a constraint is satisfied by a set of literals S, then it is satisfied by any subset of S. Our main restriction is however that there are only a fixed set of possible constraints. We show again that the satisfiability is in PTIME.

We cannot simply use the previous strategy, checking that constraints are satisfied whenever we need to apply them. The extended deduction system of the previous section is proved to be complete by a proof transformation that may not preserve constraint satisfaction. We therefore refine the strategy, memorising additional information in the formulae: on the one hand, we store the constraints that are necessarily satisfied by all instances of the clause (this is inherited in the deduction rules) and, on the other hand, the constraints that have to be satisfied in the remainder of the proofs. Using this new syntax and inference rules, we show that they do not increase the expressiveness and yet that the unit strategy is refutation complete for these new rules. This shows the PTIME membership.

In the next step, we show that the entailment problem is decidable in PTIME in this new syntax. We need however to memorise a third component, which depends on the instance of the entailment problem. This result, including constraints, is proved in section 4.4.

Final result. From the previous paragraphs, we can build a PTIME entailment algorithm which, given $S_1 \rhd t_1 \dots S_n \rhd t_n, S \rhd t$ and clauses

$$A_1 = \left\{ \begin{array}{llll} S \rhd x, & S; u_i(x) \rhd t(y) & \rightarrow & S \rhd t(y) \quad \| \quad \Gamma_i \\ S; s_j(x) \rhd v(y) & \rightarrow & S \rhd v(y) & \| \quad \Delta_j \end{array} \right.$$

where Γ_i, Δ_j are finite sets of constraints, decides in PTIME whether $S_1 \rhd t_1, \dots, S_n \rhd t_n, A_1, A_0, \mathcal{A} \models S \rhd t$.

This algorithm can be used as an oracle in a variant of the Basin and Ganzinger algorithm, to decide the satisfiability of a set of clauses including formulae extending A_0, A_1 together with ground clauses with equality. Altogether,

we obtain a PTIME procedure for arbitrary ground clauses and saturated Horn clauses (as in Basin & Ganzinger), together with the aforementioned clauses.

4.2 Tractability of deducibility axioms

We first consider the consistency problem of a very specific case:

Problem 4.1. Let $\mathcal{C}$ be a set of ground clauses built on the deducibility predicate only. Is $\mathcal{C} \cup \{\rightarrow X; x \triangleright x, \quad X \triangleright x \rightarrow X; y \triangleright x, \quad X \triangleright x,\ X; x \triangleright y \rightarrow X \triangleright y\}$ consistent?

Do note that these three axioms are the Refl, Mon, Tr axioms from the previous chapter.

Consider for instance a ground clause $a_1; \ldots; a_n \triangleright a \rightarrow \bot$. If we simply use a unit resolution strategy (which is refutation complete for Horn clauses), this single clause, together with the weakening clause, may generate all unit clauses $S \triangleright a \rightarrow \bot$ where $S \subseteq \{a_1, \ldots, a_n\}$. This should be avoided since we seek for a polynomial time algorithm. Similar problems occur with transitivity, if we try to use binary resolution with a simple strategy. Here is a more concrete example.

Example 4.2. Let $\mathcal{C} = \{a_1; a_2; a_3 \triangleright a_0 \rightarrow \bot, \quad \rightarrow a_1; a_4 \triangleright a_0, \quad \rightarrow a_2 \triangleright a_4\}$. The system $\mathcal{C} \cup \{\text{Mon}, \text{Tr}\}$ is provably unsatisfiable using binary resolution modulo ACIN only.

$$\dfrac{\dfrac{\rightarrow a_1; a_4 \triangleright a_0 \quad X_1 \triangleright x_1 \rightarrow X_1; y_1 \triangleright x_1}{\rightarrow a_1; a_4; y_1 \triangleright a_0} \qquad X_2 \triangleright x_2, \quad X_2; x_2 \triangleright y_2 \rightarrow X_2 \triangleright y_2}{a_1; y_1 \triangleright a_4 \rightarrow a_1; y_1 \triangleright a_0}$$

with unifiers $X_1 = a_1; a_4$, $X_2 = a_1; y_1$, $x_1 = a_0$, $x_2 = a_4$ and $y_2 = a_0$

$$\dfrac{a_1; y_1 \triangleright a_4 \rightarrow a_1; y_1 \triangleright a_0 \qquad \dfrac{\rightarrow a_2 \triangleright a_4 \quad X_3 \triangleright x_3 \rightarrow X; y_3 \triangleright x_3}{\rightarrow a_2; y_3 \triangleright a_4}}{\rightarrow a_1; a_2 \triangleright a_0}$$

with unifiers $X_3 = a_2$, $y_1 = a_2$ and $y_3 = a_1$

and

$$\dfrac{\dfrac{\rightarrow a_1; a_2 \triangleright a_0 \quad X_4 \triangleright x_4 \rightarrow X_4; y_4 \triangleright x_4}{\rightarrow a_1; a_2; y_4 \triangleright a_0} \qquad a_1; a_2; a_3 \triangleright a_0 \rightarrow \bot}{\bot}$$

with unifiers $X_4 = a_1; a_2$, $x_4 = a_0$ and $y_4 = a_3$.

This derivation introduces the clause $\rightarrow a_1; a_2 \triangleright a_0$, where $a_1; a_2$ is a new set (i.e. it does not appear in the initial sets). This is actually unavoidable: any derivation of the empty clause requires as an intermediate step the derivation of either $\rightarrow a_1; a_2 \triangleright a_0$ or $a_1; a_4; a_3 \triangleright a_0 \rightarrow \bot$. Both of them involve sets that are not in the initial class.

However if we move from the object level to the meta-level, viewing weakening and transitivity as inference rules and deducibility atoms as clauses, we can

at least solve this very particular case. More precisely, consider the inference system:

$$\frac{}{X; x \triangleright x}\, R \qquad \frac{X \triangleright x}{X; y \triangleright x}\, W \qquad \frac{X \triangleright x \quad X; x \triangleright y}{X \triangleright y}\, T$$

where X is a logical variable ranging over extended terms and x, y are logical variables ranging over terms.
Let $\vdash_{R,W,T}$ be the derivability relation associated with these three inference rules.

Lemma 4.3. *Given ground atomic formulae $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n$ and $S \triangleright t$, we can decide in linear time whether $\{S_1 \triangleright t_1, \ldots, S_n \triangleright t_n\} \vdash_{R,W,T} S \triangleright t$.*

Proof. We associate with each term occurring in $S_1 \cup \ldots \cup S_n \cup S \cup \{t_1, \ldots, t_n, t\}$ a proposition variable. We claim that $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n \vdash_{R,W,T} S \triangleright t$ iff $S \to t$ is derivable from $S_1 \to t_1, \ldots, S_n \to t_n$ using the propositional binary resolution, excluded middle and weakening rules only. Indeed we notice that T, R and W can be simulated by resolution and excluded middle. For W the proof rewriting is straightforward. We present the proof rewriting for T and R:

$$\frac{S \triangleright t \quad S; t \triangleright u}{S \triangleright u}\, T \quad \Longrightarrow \quad \frac{S \to t \quad S, t \to u}{S \to u}\, Res$$

$$\frac{}{S; t \triangleright t}\, R \Longrightarrow \frac{\dfrac{}{t \to t}\, Excl}{S, t \to t}\, Weak$$

Conversely the resolution, excluded middle and weakening can be simulated by R, T and W. The proof rewriting is straightforward for excluded middle and weakening, we only present it for resolution:

$$\frac{S_1 \to t \quad S_2, t \to u}{S_1, S_2 \to u}\, Res \quad \Longrightarrow \quad \frac{\dfrac{S_1 \triangleright t}{S_1; S_2 \triangleright t}\, W \qquad \dfrac{S_2; t \triangleright u}{S_1; S_2; t \triangleright u}\, W}{S_1; S_2 \triangleright u}\, T$$

Then derivability of $S \to t$ is equivalent to unsatisfiability of $S_1 \to t_1, \ldots, S_n \to t_n, S, \neg t$ (where S_i is a shortcut for the conjunction of propositional variables corresponding to terms occurring in S_i), which can be decided in linear time: it is a HornSat problem. □

Now, the trick of viewing the clauses Mon, Tr as new inference rules allows to decide our problem in PTIME. We write $\vdash_{Res_u+R+W+T}$ for the derivability with inference rules R, W, T and unit resolution.

Lemma 4.4. *Given a set of ground Horn clauses (built on $\triangleright$) $\mathcal{C}$, the satisfiability of $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\}$ is decidable in cubic time.*

Proof. We show first that $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\}$ is unsatisfiable if and only if the empty clause can be derived from $\mathcal{C}$, using unit resolution $R + W + T$. If we can derive the empty clause in this system, then we can derive the empty clause from $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\}$ by resolution, thanks to simple proof rewriting rules:

$$\frac{}{S; t \triangleright t}\, R \quad \Longrightarrow \quad S; t \triangleright t \text{ (instance of Refl)}$$

$$\dfrac{\dfrac{\pi_1}{S \rhd t}}{S; u \rhd t}\, W \quad \Longrightarrow \quad \dfrac{\dfrac{\pi_1}{S \rhd t} \qquad X \rhd x \ \rightarrow\ X; y \rhd x}{S; u \rhd t}\, Res$$

$$\dfrac{\dfrac{\pi_1}{S \rhd t} \qquad \dfrac{\pi_2}{S; t \rhd u}}{S \rhd u}\, T \quad \Longrightarrow \quad \dfrac{\dfrac{\dfrac{\pi_1}{S \rhd t} \qquad X; x \rhd y,\ X \rhd x \ \rightarrow\ X \rhd y}{S; t \rhd y \rightarrow S \rhd y}\, Res \qquad \dfrac{\pi_2}{S; t \rhd u}}{S \rhd u}\, Res$$

Conversely, if we cannot derive the empty clause from $\mathcal{C}$ using unit resolution $R + W + T$, then let $\mathcal{M} = \{S \rhd u \mid \mathcal{C} \vdash_{Res_u+R+W+T} S \rhd u\}$. We claim that $\mathcal{M}$ is a model of $\mathcal{C} \cup \{\textsc{Refl}, \textsc{Mon}, \textsc{Tr}\}$: As $\mathcal{M}$ is closed by R, W, T, it is a model of $\{\textsc{Refl}, \textsc{Mon}, \textsc{Tr}\}$ and, if $B_1, \ldots, B_n \rightarrow H \in \mathcal{C}$, then either $B_i \notin \mathcal{M}$ for some i or else, by construction, for every i, $\mathcal{C} \vdash_{Res_u+R+W+T} B_i$, hence, by unit resolution, $\mathcal{C} \vdash_{Res_u+R+W+T} H$. In all cases, $\mathcal{M} \models B_1, \ldots, B_n \rightarrow H$.

It only remains to prove that whether $\mathcal{C} \vdash_{Res_u+R+W+T} \bot$ or not can be decided in cubic time. Let $\mathcal{B}$ be the set of atomic formulae occurring in $\mathcal{C}$. Let $\mathcal{M}$ be the least fixed point of

$$f(X) = \{S \rhd u \in \mathcal{B} \mid \mathcal{C} \cup X \vdash_{Res_u} S \rhd u \text{ or } \mathcal{C} \cup X \vdash_{R+W+T} S \rhd u\}$$

Since f is monotone, there is a least fixed point, which is contained in $\mathcal{B}$ by construction. Computing $\mathcal{M}$ can be performed in quadratic time, as there are at most $|\mathcal{B}|$ iterations and each step requires at most a linear time, thanks to the Lemma 4.3.

If the empty clause can be derived from $\mathcal{M}, \mathcal{C}$ using unit resolution, then $\mathcal{C} \vdash_{Res_u+R+W+T} \bot$. Let us show the converse implication. For this, we prove, by induction on the proof size that, for every atomic formula $S \rhd t \in \mathcal{B}$, $\mathcal{C} \vdash_{Res_u+R+W+T} S \rhd t$ implies $S \rhd t \in \mathcal{M}$.

If the last rule of the proof is a unit resolution, then the proof can be written:

$$\dfrac{\dfrac{\pi_1}{S_1 \rhd t_1} \qquad \dfrac{\dfrac{\pi_2}{S_2 \rhd t_2} \qquad \dfrac{\dfrac{\dfrac{\pi_n}{S_n \rhd t_n} \qquad \dfrac{}{S_1 \rhd t_1, \ldots, S_n \rhd t_n \rightarrow S \rhd t}\, {\scriptstyle (S_1 \rhd t_1, \ldots, S_n \rhd t_n \rightarrow S \rhd t) \in \mathcal{C}}}{S_1 \rhd t_1, \ldots, S_{n-1} \rhd t_{n-1} \rightarrow S \rhd t}}{\vdots}}{S_1 \rhd t_1, S_2 \rhd t_2 \rightarrow S \rhd t}}{S_1 \rhd t_1 \rightarrow S \rhd t}}{S \rhd t}$$

$S_1 \rhd t_1, \ldots, S_n \rhd t_n \in \mathcal{B}$ and, by induction hypothesis, $S_1 \rhd t_1, \ldots, S_n \rhd t_n \in \mathcal{M}$. It follows that $\mathcal{M}, \mathcal{C} \rhd_{Res_u} S \rhd t$, hence $S \rhd t \in f(\mathcal{M}) = \mathcal{M}$.

If the last rule of the proof is W or T, then there are atomic formulae $S_1 \rhd t_1, \ldots, S_n \rhd t_n$ such that $S_1 \rhd t_1, \ldots, S_n \rhd t_n \vdash_{R+W+T} S \rhd t$ and, for every i, either $S_i \rhd t_i \in \mathcal{C}$ or the last rule in the proof of $S_i \rhd t_i$ is a resolution step and, as noticed previously all, $S_i \rhd t_i$ are in $\mathcal{B}$. In all cases $S_i \rhd t_i \in \mathcal{B}$ and, by induction hypothesis, $S_i \rhd t_i \in \mathcal{M}$. By definition of the function f, $S \rhd t \in f(\mathcal{M}) = \mathcal{M}$.

If $\mathcal{C} \vdash_{Res_u+R+W+T} \bot$, then there is a negative clause $S_1 \rhd t_1, \ldots, S_n \rhd t_n \rightarrow \bot$ in $\mathcal{C}$ such that, for every i, $\mathcal{C} \vdash_{Res_u+R+T+W} S_i \rhd t_i$, hence $S_i \rhd t_i \in \mathcal{M}$ as we

just saw. Then $\bot$ can be deduced from $\mathcal{C}, \mathcal{M}$ using unit resolution (which can be decided in linear time again). □

Example 4.5. Let us get back to to Example 4.2:

$$\mathcal{C} = \{a_1; a_2; a_3 \triangleright a_0 \to \bot, \quad \to a_1; a_4 \triangleright a_0, \quad \to a_2 \triangleright a_4\}$$

There is no unit resolution step on $\mathcal{C}$. We claim that $\mathcal{C} \vdash_{R+W+T} a_1; a_2; a_3 \triangleright a_0$. Indeed, turning atoms into clauses, from the clauses $\{a_1; a_4 \to a_0, \quad a_2 \to a_4, \to a_1, \to a_2, \ a_0 \to \bot\}$ we deduce $\bot$ by unit resolution. Hence $a_1; a_2; a_3 \triangleright a_0$ belongs to $f(\emptyset)$. Then, by one step of unit resolution on $\mathcal{C} \cup \{a_1; a_2; a_3 \triangleright a_0\}$, we get a contradiction.

$$\frac{}{X; x \triangleright x}\, R \qquad \frac{X \triangleright x}{X; y \triangleright x}\, W \qquad \frac{X \triangleright x \quad X; x \triangleright y}{X \triangleright y}\, T$$

Figure 4.1: Inference rules for Refl, Mon, Tr without equalities

4.2.1 Adding equality

Now, we assume that atomic formulae in $\mathcal{C}$ may contain equalities on terms (not extended terms). The equality axioms (the equality is a congruence) are implicit in what follows.

We start by adding inference rules as previously, in order to cope with the equalities. We define the unit resolution rule with equalities $Res_u(E)$ as follows, where the predicate p ranges over $=$ and $\triangleright$

$$\frac{p(t_1, t_2) \quad p(u_1, u_2), A_1, \cdots, A_n \to B}{A_1, \cdots, A_n \to B}\, t_1 =_E u_1, t_2 =_E u_2$$

We also define an equality elimination rule $\text{Elim}(E)$ as follows:

$$\frac{t_1 = t_2, A_1, \cdots, A_n \to B}{A_1, \cdots, A_n \to B}\, t_1 =_E t_2$$

We modify the transitivity rule as follows:

$$\frac{X \triangleright x \quad X; z \triangleright y \quad x =_E z}{X \triangleright y}\, T(E)$$

Lemma 4.6. *Given a list of terms $S_1 \triangleright u_1, \cdots, S_n \triangleright u_n$, a finite set of ground equations E and a goal term $S \triangleright u$, the problem $E, S_1 \triangleright u_1, \cdots, S_n \triangleright u_n, (x \triangleright x)_{x \in \mathcal{T}} \vdash^?_{W,T(E)} S \triangleright u$ is decidable in polynomial time with access to an oracle deciding E.*

Proof. Let T be the set of terms occurring in $S_1, \ldots, S_n, S, u_1, \ldots, u_n, u$. Split T into disjoint equivalence classes modulo E, calling (at most) a quadratic number of times the oracle deciding E. For each equivalence class, choose a representative and replace the terms in $S_1 \triangleright u_1, \cdots, S_n \triangleright u_n, S \triangleright u$ with their representatives. Then, the resulting entailment problem can be turned into a HornSat problem (as before), replacing every representative of an equivalence class with a proposition variable. □

Lemma 4.7. *Given a set of ground Horn clauses (built on $\triangleright$ and $=$) $\mathcal{C}$, the satisfiability of $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\}$ is decidable in polynomial time.*

Proof. Let $\mathcal{B}_1$ be the set of equalities occurring in $\mathcal{C}$. Let $\mathcal{B}_2$ be the set of $S \triangleright t$ occurring in $\mathcal{C}$, as well as $\bot$.

We want to compute the least fixed point of the following function F which takes as input a set of equations $E \subseteq \mathcal{B}_1$ and a set of $\triangleright$ literals $\mathcal{B}$ and returns E' and $\mathcal{B}'$ built as follows:

$$\mathcal{B}' = \{S \triangleright t \in \mathcal{B}_2 | \mathcal{C} \cup \mathcal{B} \vdash_{Res_u(E),\text{Elim}(E)} S \triangleright t\} \cup \{S \triangleright t \in \mathcal{B}_2 | \mathcal{B} \vdash_{R,W,T(E)} S \triangleright t\}$$

$$E' = \{u = v \in \mathcal{B}_1 | \mathcal{C} \cup \mathcal{B} \vdash_{Res_u(E),\text{Elim}(E)} u = v\}$$

Our algorithm answers Unsatisfiable iff $\bot$ is derivable using from the least fixed point of F using unit resolution.

E is always a finite (polynomially bounded) set of ground equations. Hence there is an polynomial time oracle that decides the equality modulo E, for instance using a congruence closure algorithm. Then, thanks to lemma 4.6, F can be computed in polynomial time. Furthermore, the number of iterations of F is linear. Hence the fixed point can be computed in polynomial time.

Let $\mathcal{B}_f, E_f$ be the least fixed point of F. Let us prove now that $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\}$ is satisfiable iff $\bot \notin \mathcal{B}_f$.

If $\bot \in \mathcal{B}_f$, then $\mathcal{C} \cup \{r, w, t\}$ is unsatisfiable since every deduction step used in the computation of F is a consequence of $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\}$ (and the equality axioms).

Conversely, if $\bot \notin \mathcal{B}_f$, we consider the first-order structure $\mathcal{M}$, in which the interpretation domain is the quotient $\mathcal{T}/=_{E_f}$ of the set of ground terms by the congruence generated by E_f and the interpretation of $\triangleright$ is the set $\{S \triangleright t \mid \mathcal{B}_2 \vdash_{R,W,T(E_f)} S \triangleright t\}$.

$\mathcal{M}$ is, by construction, a model of Refl, Mon, Tr. If $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n, u_1 = v_1, \ldots, u_m = v_m \rightarrow H$ is a clause of $\mathcal{C}$, and, for every i, $\mathcal{M} \models S_i \triangleright t_i$ and, for every j, $u_j =_{E_f} v_j$, then, for every i, $\mathcal{B}_f \vdash_{R,W,T(E_f)} S \triangleright t_i$. Hence $S_i \triangleright t_i$ is in the first component of $F(\mathcal{B}_f, E_f)$, hence in $\mathcal{B}_f$.

It follows that, $\mathcal{B}_f, \mathcal{C} \vdash_{Res_u(E),\text{Elim}(E)} H$. Hence $H \neq \bot$ and H is in either components of $F(\mathcal{B}_f, E_f) = (\mathcal{B}_f, E_f)$. Therefore $\mathcal{M} \models H$.

We have proved that $\mathcal{M}$ is a model of each clause of $\mathcal{C}$. Since it is a model of Refl, Mon, Tr, this concludes the proof. □

$$\frac{}{X; x \triangleright x}\, R \qquad \frac{X \triangleright x}{X; y \triangleright x}\, W \qquad \frac{X \triangleright x \quad X; x \triangleright y \quad x =_E z}{X \triangleright y}\, T(E)$$

Figure 4.2: Inference rules for Refl, Mon, Tr with equalities

4.2.2 Adding a function axiom

We extend now the clauses specifying $\triangleright$ with the functionality axioms for $\mathcal{F}$ (denoted by Fun($\mathcal{F}$) later): $X \triangleright x_1, \quad \cdots \quad X \triangleright x_n \quad \rightarrow \quad X \triangleright g(x_1, \ldots, x_n)$, for every function symbol g in a set of function symbols $\mathcal{F}$ (which is later omitted).

We will use, as the main tool of this proof, the fact that it is possible to decide whether, given $t_1, \ldots, t_n, u$ and E, there exists a context C such that $C[t_1, \ldots, t_n] =_E u$. Using that tool, we will be able to extend the inference rules to accommodate the functionality axioms. We try to give the intuition in the next example:

Example 4.8. $b \rhd c, \rhd a \vdash_{R+W+T(g(g(a))=b)+F_g} \rhd c$ since there is a context C (with $C[\_] = g(g(\_))$) such that $C[a] = b$.

First, note that the axiom $\textsc{Fun}_f$: $S \rhd x_1, \cdots, S \rhd x_n \to S \rhd f(x_1, \cdots, x_n)$ is equivalent (modulo weakening and transitivity) to $t_1; \ldots; t_{n_f} \rhd f(t_1, \cdots, t_{n_f})$

Now we know, as a consequence of Lemma 4.7 that the set of clauses $\mathcal{C}$ is satisfiable together with the axioms $\{\textsc{Refl}, \textsc{Mon}, \textsc{Tr}\} \cup \textsc{Fun}(\mathcal{F})$ iff the empty clause is not derivable from $\mathcal{C}$ together with all possible instances of the function axiom $\{t_1; \ldots; t_{n_f} \rhd f(t_1, \cdots, t_{n_f}) | f \in \mathcal{F}, t_1, \ldots, t_{n_f} \in \mathcal{T}\}$ using the rules $T(E)$, $Res_u(E)$, $\textsc{Elim}(E)$ and W.

Lemma 4.9. *Given a set of ground equations E, a set of ground terms $u, t_1, \ldots, t_n$, and $\mathcal{F}$ a set of function symbols. The problem $\exists C. C[t_1, \ldots, t_n] =_E u$ with C (multi)context built on $\mathcal{F}$ is decidable in polynomial time.*

Proof. We proceed as follows (the steps will be precised later):

1. Build a tree automaton $\mathcal{A}$ (of polynomial size) that recognises the set of all t such that $t =_E u$.
   (a) Compute (in polynomial time in $|E|$) a flat convergent rewriting system R for E (of polynomial size in the size of E).
   (b) Build a tree automaton (of polynomial size in $|R| + |t|$), which accepts the terms that rewrite to $t{\downarrow_R}$.
2. Build a tree automaton $\mathcal{B}$ (of polynomial size in $\Sigma_i |t_i|$) that recognises the language $\{C[t_1, \ldots, t_n]\}$.
3. Check (in polynomial time in $|\mathcal{A}| + |\mathcal{B}|$) whether $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$.

1a - We add a constant c_u for every subterm u of E, t and add an equation $c_u = u$ to E. In this way, we may now assume w.l.o.g that every equation in E has the form $f(a_1, \ldots, a_n) = a$ or $a_1 = a$ (we call equations of this form flat). We choose an arbitrary linear order on symbols in which the non-constant function symbols are greater than the constants and run a Knuth-Bendix completion on E using a lexicographic path ordering that extends this precedence. This yields a flat convergent rewrite system R whose size is polynomial in E. Indeed the superposition of two flat equations is necessarily flat, and there are only polynomially many such equations. This requires only a polynomial time.

1b - We want to recognise the set of terms u such that $u{\downarrow_R} = t{\downarrow_R}$. Note that $t{\downarrow_R}$ is a constant c_t. Now build a tree automaton $\mathcal{A}$ as follows:

- the set of states of $\mathcal{A}$ is the set S of constants appearing in R,
- for each constant c add a transition $c() \to c$
- for each rule $f(a_1, \ldots, a_n) \to a$ in R add a transition $f(a_1, \ldots, a_n) \to a$ in $\mathcal{A}$
- for each rule $a_1 \to a$ in R and every transition $f(a_1, \ldots, a_n) \to a_1$ replace a_1 by a in the transition (applying this point starting from the highest a_1 in the order chosen to complete E).
- the accepting state of $\mathcal{A}$ is c_t

Note that this procedure yields a polynomial size $\mathcal{A}$ in polynomial time. If $\mathcal{A}$ recognises u, it is clear that the accepting run of $\mathcal{A}$ can be seen as a rewrite sequence from u to c_t. Conversely, each rewrite sequence from u to c_t yields an accepting run of $\mathcal{A}$ on u.

2 - Build the tree automaton $\mathcal{A}_1, \ldots, \mathcal{A}_n$ recognising the terms $t_1, \ldots, t_n$ with accepting state q_0. Now let $\mathcal{A}'$ be the automaton recognising the language $t_1, \ldots, t_n$ with accepting state q_0 (it is the sum of the n previous automata). Let $\mathcal{B}$ be the automaton obtained extending $\mathcal{A}'$ with the transitions $f(q_0, \ldots, q_0) \to q_0$. Note that $\mathcal{B}$ is built in polynomial time and is of polynomial size. It is clear that $\mathcal{B}$ recognises the language $\{C[t_1, \ldots, t_n] |\!|\ C \text{ context built on } \mathcal{F}\}$.

3 - Build the product automaton that recognises $L(\mathcal{A}) \cap L(\mathcal{B})$ (of polynomial size) and test for emptiness in polynomial time.

□

Now having proven that we can decide whether a term t can be built from a set of terms u_i, we are left with proving that entailment with the transitivity and monotonicity rules is still decidable in **PTIME**, which is the fundamental tool for building the decision procedure.

Lemma 4.10. *Given a list of terms $S_1 \rhd u_1, \cdots, S_n \rhd u_n$, a finite set of ground equations E and a goal term $S \rhd u$, the problem $S_1 \rhd u_1, \cdots, S_n \rhd u_n, (t_1; \ldots; t_{n_f} \rhd f(t_1, \cdots, t_{n_f}))_{f \in \mathcal{F}, t_1, \ldots, t_{n_f} \in \mathcal{T}} \vdash^?_{W,T(E)} S \rhd u$ is decidable in polynomial time.*

Proof. Note that in the proof of lemma 4.6 we saturate $S_1 \to u_1 \ldots S_n \to u_n, S$, modulo the unit version of $T(E)$ and check if we obtain u. Now, we need to saturate $S_1 \to u_1 \ldots S_n \to u_n, S, \neg u, (f(t_1, \cdots, t_{n_f}))_{f \in \mathcal{F}, t_1, \ldots, t_{n_f} \in \mathcal{T}}$ modulo the unit version of $T(E)$. Observe the following: if a function clause is used to derive u then it is used in a proof that has the following structure (we omit here that everything is done modulo E)

$$\dfrac{t_i \qquad \dfrac{\dfrac{t_l \qquad t_1, \ldots, t_n \to f(t_1, \ldots, t_n)}{t_1, \ldots, t_{l-1}, t_{l+1}, \ldots, t_n \to f(t_1, \ldots, t_n)} \quad \cdots}{t_i \to f(t_1, \ldots, t_n)}}{f(t_1, \ldots, t_n)} \qquad f(t_1, \ldots, t_n), A_1, \ldots, A_k \to B}{A_1, \ldots, A_k \to B}$$

In its turn either t_i is a term in $U = \bigcup_i S_i \cup S \cup \{u_1, \ldots, u_n\}$ or its proof has the structure shown above. Therefore, there exists $v_1, \ldots, v_l, w \in U$ (and the units $v_1, \ldots, v_l$ are derivable) such that $w = f(t_1, \ldots, t_n)$ and $C[v_1, \ldots, v_l] =_E w$. Note that this observation gives us the following:

$$E, \quad S_1 \rhd u_1, \cdots, S_n \rhd u_n, \\ (t_1; \ldots; t_{n_f} \rhd f(t_1, \cdots, t_{n_f}))_{f \in \mathcal{F}, t_1, \ldots, t_{n_f} \in \mathcal{T}} \vdash^?_{W,T(E)} S \rhd u$$

is decidable in PTIME by saturating $E, S_1 \to u_1, \cdots, S_n \to u_n, S$ by

$$\dfrac{x_1 \qquad \ldots \qquad x_n \qquad X, z \to y}{X \to y} \; \exists C. C[x_1, \ldots, x_n] =_E z$$

and checking whether $\exists C.C[v_1, \ldots, v_k] =_E u$ where $v_1, \ldots, v_k$ are the units derived by the saturation. As checking the condition $\exists C.C[t_1, \ldots, t_n] =_E u$ is decidable in PTIME, the saturation is in PTIME. □

As previously, we deduce from the existence of a decision procedure for the rule operating only on the $\triangleright$ atoms a decision procedure for the complete problem.

Lemma 4.11. *Given a set of ground Horn clauses (built on $\triangleright$ and $=$) $\mathcal{C}$, the satisfiability of $\mathcal{C} \cup \{\text{Refl}, \text{Mon}, \text{Tr}\} \cup \text{Fun}(\mathcal{F})$ is decidable in polynomial time.*

Proof. The proof goes exactly as the proof of lemma 4.7 except that we use the oracle of lemma 4.10 instead of the oracle of lemma 4.6. □

$$\frac{}{X; x \triangleright x}\, R \qquad \frac{X \triangleright x}{X; y \triangleright x}\, W$$

$$\frac{x_1 \quad \ldots \quad x_n \quad X; x \triangleright y \quad \exists C.C[x_1, \ldots, x_n] =_E x}{X \triangleright y}\, T(E)$$

Figure 4.3: Inference rules for Refl, Mon, Tr, Fun($\mathcal{F}$)

4.3 More clauses using the deducibility predicate

We now enrich the class of clauses involving the deducibility predicate. Given a linear term p (later called the *pattern*), we consider a finite set of clauses of the following forms:

$c_s(u)$: $X; u \triangleright p \rightarrow X \triangleright p$ where u is a linear term that does not share variables with p

$c_c(w)$: $X \triangleright y, X; w \triangleright p \rightarrow X \triangleright p$ where w is a linear term that does not share variables with p, and y is a variable of w.

The restriction to linear terms (i.e. terms where each variable appears at most once) might seems odd but it ensures that checking checking that a term t matches a pattern modulo an equational theory can be solved in PTIME.

Example 4.12. The secrecy axiom described in chapter 3 (stripped from the constraints)

$$X; \text{enc}(x, pk) \triangleright n \quad \rightarrow \quad X \triangleright n$$

is an instance of the first class of clauses above, with $p = n$ and $u = \text{enc}(x, pk)$. The condition usable(sk, X) requires constraints, that are considered in Section 4.4.

As explained in the previous section, we may turn the additional clauses into new inference rules, using $\leq_E$, the matching modulo E (a term t satisfies $u \leq_E t$ if there is a substitution σ such that $t =_E u\sigma$).

$$\frac{u \leq_E x \quad X; x \triangleright p}{X \triangleright p}\, \mathsf{Str}_u \qquad \frac{(y, w) \leq_E (x, z) \quad X \triangleright x \quad X; z \triangleright p}{X \triangleright p}\, \mathsf{Cut}_w$$

Let $\mathcal{I}$ be the inference system defined by a finite collection of rules Str_u, Cut_w, the rules $R, W, T(E)$ for a finite set of ground equations E and the rules F_g for a set of function symbols g.

We are going to prove that, again, $\mathcal{I}$ can be decided in polynomial time. However, we cannot use the same proof as in the previous section. $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n \vdash_{\mathcal{I}} S \triangleright t$ can no longer be reduced to a problem $S_1 \to t_1, \ldots, S_n \to t_1, S \vdash_{Res_u} t$ (modulo a PTIME oracle).

Example 4.13. Assume E is empty and we have a single rule $\mathsf{Cut}_{f(x,k)}$ for the pattern $p = n$. $f(a,k) \triangleright f(b,k),\ f(b,k) \triangleright n \vdash_{\mathcal{I}} a \triangleright n$:

$$\dfrac{\dfrac{}{a \triangleright a}\,R \qquad \dfrac{\dfrac{f(a,k) \triangleright f(b,k)}{a; f(a,k) \triangleright f(b,k)}\,W \qquad \dfrac{f(b,k) \triangleright n}{a; f(a,k); f(b,k) \triangleright n}\,W}{a; f(a,k) \triangleright n}\,T}{a \triangleright n}\,\mathsf{Cut}_{f(x,k)}$$

We cannot use a unit version of T (or resolution) in this example. And moving to a general binary resolution would yield an exponential procedure.

As before, after turning the clauses into inference rules, we turn the deducibility atomic formulae into clauses. We call again $\mathcal{I}$ the resulting inference system. We have to be careful however: this is a purely syntactic transformation and the inference rules resulting from this translation are no longer correct in a classical semantics. For instance Cut_w becomes

$$\dfrac{A_1, \ldots, A_n \to y \qquad w, B_1, \ldots, B_m \to p}{A_1, \ldots, A_n, B_1, \ldots, B_m \to p}$$

where the premises are matched modulo a set of ground equations E.

In order to apply a simple fixed point computation, we would like to be able to transform any proof into a unit strategy proof. Since this is not possible with the current proof system (as shown by Example 4.13), we introduce additional inference rules that will allow such a strategy, however bookkeeping what the rest of the proof owes, in order to enable a translation back into the original proof system.

Example 4.14. Continuing Example 4.13, the unit proof of $\to n$ from the hypotheses $\to a,\ f(a,k) \to f(b,k),\ f(b,k) \to n$ will look like this:

$$\dfrac{\dfrac{\to a \qquad f(a,k) \to f(b,k)}{\to_p f(b,k)}\,\mathsf{Cut}^1_{f(x,k)} \qquad f(b,k) \to n}{\to n}\,\mathsf{Cut}^2$$

The rule Cut^1_w is a generalisation of Cut_w since the constraint of being an instance of the pattern p on the right is dropped. It bookkeeps however a duty as a mark p on the arrow. The mark on a clause $S \to_p t$ can in turn be erased only when a clause $S', t \to p$ is one of the premises. Such a mechanism allows both to use a complete unit strategy and to enable reconstructing an original proof from the extended one, as we will prove (here the annotation is erased in the last rule as the second premise is an instance of $S, f(x,k) \triangleright n$).

Intuitively, the head s of a marked clause can only be used in a proof that will end up deriving an instance of the pattern.

We extend the syntax, allowing both unmarked clauses $S \to t$ and marked clauses $S \to_p t$. The clause $S \to_p t$ can only be used in proofs that will end up deriving an instance of p. For simplicity, we first do not consider the set of ground equations E nor the function axioms. We write $S \to_? t$ when it does not matter whether the arrow is marked or not. We then consider the inference system $\mathcal{J}$ consisting of $T(E)$, W and the following rules (for each Cut_w there are two rules Cut^i_w and for each rule Str_u there are two rules Str^i_u):

$$\frac{A_1, \ldots, A_n \to_? y \qquad B_1, \ldots, B_m, w \to_? v}{A_1, \ldots, A_n, B_1, \ldots, B_m \to_p v}\,\mathsf{Cut}^1_w$$

$$\frac{A_1, \ldots, A_n \to_? y \qquad w, B_1, \ldots, B_m \to p}{A_1, \ldots, A_n, B_1, \ldots, B_m \to p}\,\mathsf{Cut}^2_w$$

$$\frac{A_1, \ldots, A_n \to_? x \qquad B_1, \ldots, B_m, x \to_? v}{A_1, \ldots, A_n, B_1, \ldots, B_m \to_? v}\,\mathsf{Cut}^1$$

in which the conclusion is marked iff one of the premises is marked.

$$\frac{A_1, \ldots, A_n \to_? x \qquad x, B_1, \ldots, B_m \to p}{A_1, \ldots, A_n, B_1, \ldots, B_m \to p}\,\mathsf{Cut}^2$$

$$\frac{A_1, \ldots, A_n, u \to_? x}{A_1, \ldots, A_n \to_p x}\,\mathsf{Str}^1_u \qquad \frac{A_1, \ldots, A_n, u \to p}{A_1, \ldots, A_n \to p}\,\mathsf{Str}^2_u$$

Note that the above system has no classical semantics.

Lemma 4.15. *Let $\mathcal{S}$ be a set of ground clauses, and s be a ground term. In case $E = \emptyset$ and removing the function and reflexivity axioms from $\mathcal{I}$, $\mathcal{S} \vdash_{\mathcal{I}} \to s$ if and only if $\mathcal{S} \vdash_{\mathcal{J}} \to s$.*

Let us first give a proof sketch of this result before diving into the full proof. *Proof sketch*: For one implication we prove that W is not necessary, hence $\mathcal{I}$ can be simulated by $\mathcal{J}$. For the other implication, we rewrite a proof in $\mathcal{J}$ as follows. We consider a last rule that introduces a mark. Since the marks must eventually disappear, there is also a matching rule that removes the mark. This part of proof is then rewritten as explained on the following example:

$$\dfrac{\dfrac{S_n \to t_n \qquad \vdots}{S'_n \to_p v\sigma}\,\mathsf{Cut}^1_{w_n} \qquad \dfrac{\dfrac{S_2 \to t_2 \qquad \dfrac{S_1 \to t_1 \quad S, w\sigma \to v\sigma}{S_1, S \to_p v\sigma}\,\mathsf{Cut}^1_w}{S'_2 \to_p v\sigma}\,\mathsf{Cut}^1_{w_2}}{S_0, t\sigma' \to p\theta}\,\mathsf{Cut}^2_t}{S_0, S'_n \to p\theta}$$

rewrites to

$$\dfrac{S_n \to t_n \qquad \vdots}{S_0, S'_n \to p\theta}\,\mathsf{Cut}_{w_n}$$

$$\dfrac{S_1 \to t_1 \qquad \dfrac{S, w\sigma \to v\sigma \quad S_0, t\sigma' \to p\theta}{S_0, S, w\sigma \to p\theta}\,\mathsf{Cut}_t}{S_0, S_1, S \to p\theta}\,\mathsf{Cut}_w$$

The proof rewriting terminates and we end up with a proof in $\mathcal{I}$. Let us prove this result formally.

Proof. We first prove that, if there is a proof Π of s in $\mathcal{I}$ from $\mathcal{S}$, then there is a proof Π' without W. Indeed, we may push W to the bottom of the proof as follows:

$$\dfrac{\dfrac{A_1, \ldots, A_n \to x}{A_1, \ldots, A_n, C \to x}\, W \qquad B_1, \ldots, B_m, w \to p}{A_1, \ldots, A_n, B_1, \ldots, B_m, C \to p}\, \mathsf{Cut}_w$$

can be rewritten to

$$\dfrac{A_1, \ldots, A_n \to x \quad B_1, \ldots, B_m, w \to p}{\dfrac{A_1, \ldots, A_n, B_1, \ldots, B_m, \to p}{A_1, \ldots, A_n, B_1 \ldots, B_m, C \to p}\, W}\, \mathsf{Cut}_w$$

W also commutes with the rules Str_u. Since the proof of a unit clause cannot end with W, Π does not contain W.

Now let us show that if there is a proof of $\to s$ in $\mathcal{J}$ then there is a proof of $\to s$ in $\mathcal{I}$: Consider a minimal (in number of Cut^1, Cut^1_w, Str^1_u rules) proof Π of $S \to t$ in $\mathcal{J}$. Consider a subproof Π' of Π that uses once Cut^2_w, as a last inference rule. We show that Π' can be rewritten into a strictly smaller proof (w.r.t. the size). This contradicts the minimality of Π, hence this proves that the minimal size proof does not make use of any extra rule.

First note that, according to labels inheritance, once a clause is annotated, then the label cannot be removed completely, unless we apply Cut^2_w or Cut^2. Since the leaves of Π' are not annotated, we can write Π' as:

$$\dfrac{\dfrac{\dfrac{\vdots\ \pi_1}{S^1 \to t}\, R^1 \qquad \vdots}{S^n \to_p t}\, R^n \qquad \pi_2}{S^n, S \to p\sigma}\, \mathsf{Cut}^2_w \qquad \begin{array}{c} \\ S, w\sigma \to p\sigma \end{array}$$

where $R^1, \ldots, R^n$ are Cut^1_w, Cut^1 or Cut^1_u.

We argue that Π' can be rewritten into

$$\dfrac{\dfrac{\vdots}{\dfrac{\dfrac{\pi_1}{S^1 \to t} \qquad \dfrac{\pi_2}{S, w\sigma \to p\sigma}\, \mathsf{Cut}^2_w}{S^1, S \to p\sigma}\, \widetilde{R^1}} \qquad \vdots}{S^n, S \to p\sigma}\, \widetilde{R^n}$$

This proof contains a strictly less annotations. It only remains to define the rules $\widetilde{R^i}$ and check that the above proof is a valid proof in the new inference system indeed. Let $\bar{\sigma}$ be σ restricted to the variables of p. As the variables of p are disjoint from those of u, w, the following $\widetilde{R_k}$ rules are well defined.

$$\text{If} \qquad R^k = \frac{V_2^k \ \to_p t^k \quad V_1^k, w'\sigma' \ \to_p t}{S^k \ \to_p t}$$

$$\text{We let} \qquad \widetilde{R_k} = \frac{V_2^k \ \to_p t^k \quad S, V_1^k, w'\sigma'\bar{\sigma} \to p\sigma'\bar{\sigma}}{S, S^k \to p\sigma'\bar{\sigma}}$$

The rule $\mathsf{Cut}^1_{w'}$ is therefore replaced with a rule $\mathsf{Cut}^2_{w'}$.

$$\text{If} \qquad R^k = \frac{V_1^k, v\sigma' \to_p t}{V_1^k \to_p t} \qquad \text{we let} \qquad \widetilde{R^k} = \frac{S, V_1^k, v\sigma'\bar{\sigma} \to p\sigma'\bar{\sigma}}{S, V_1^k \to p\sigma'\bar{\sigma}}$$

The rule Str^1_v is replaced with a rule Str^2_v.

It is now enough to note that the choice of $\widetilde{R^k}$ ensures that Π' is a valid proof in the $\mathcal{I}$ inference system. □

The *unit* strategy for $\mathcal{J}$ consists in applying the rules only when $n = 0$ for the Cut^i_w rules (i.e. when the left premise of a Cut^i_w is a unit clause).

Lemma 4.16. *If $\mathcal{S} \vdash_{\mathcal{J}} \to s$ then $\to s$ is derivable from $\mathcal{S}$ in $\mathcal{J}$ using the unit strategy.*

Proof. Let us first sketch the proof. We prove this lemma by induction on the proof size. We assume w.l.o.g. that all proofs of literals (whether marked or not) labelling a node in the proof (except the root) use a unit strategy. We consider the last step that does not comply with the unit strategy. If $A_1, \ldots, A_n \to_? s$ is its conclusion, then all atoms $A_1, \ldots, A_n$ can be proved in $\mathcal{J}$ with the unit strategy. We therefore simplify the premises accordingly, which yields an inference rule complying with the unit strategy.

In more details, let Π be a proof of $\to s$ in $\mathcal{J}$ minimal in the number of non unit cuts. Assume, by contradiction that Π uses at least one non-unit rule, for example the following instance of Cut^2_w,

$$R^0 \frac{S \to_p u \quad S', w\sigma \to p\sigma}{S, S' \to p\sigma}$$

then as the conclusion of Π is a unit clause, Π has a subproof of the following form:

$$\dfrac{\dfrac{\dfrac{}{S^0 \to p\sigma}\, R^0}{S^1 \to p\sigma}\, R^1}{\vdots}$$
$$\dfrac{}{\to p\sigma}\, R^n$$

Let $I = \{i_1, \ldots, i_l\}$ be the set of indices such that $S^{i-1} \backslash S^i \subseteq S$. If $i \in I$ and

$$R^i \frac{\to_? t^i \quad S^{i-1} \to p\sigma}{S^i \to p\sigma}$$

we let

$$\widetilde{R^i}\frac{\rightarrow_? t^i \quad S \cap S^{i-1} \rightarrow_p u}{S \cap S^i \rightarrow_p u}$$

and if $i \in I$ and

$$R^i\frac{S^{i-1} \rightarrow p\sigma}{S^i \rightarrow p\sigma}$$

we let

$$\tilde{R^i}\frac{S^{i-1} \rightarrow_p u}{S^i \rightarrow_p u}$$

Then replacing the original subproof by the following one in Π yields a proof with one less non-unit cut.

$$\dfrac{\dfrac{S \rightarrow_p u}{\rightarrow_p u}\widetilde{R^{i_1}} \quad \dfrac{\cdots}{}\widetilde{R^{i_l}} \qquad S', w\sigma \rightarrow p\sigma}{S' \rightarrow p\sigma}$$

□

As previously, solving the entailment problem on $\triangleright$ atoms is the key to proving the more general Theorem 3.

Theorem 3. *If $\mathcal{S}$ is a set of ground clauses built on $\triangleright$, we can decide in PTIME the satisfiability of $\mathcal{S}$, together with T, W and finitely many clauses c_s, c_c, that are built on the same pattern p.*

Proof. First observe that the unit resolution strategy in 4.16 yields a PTIME decision procedure for the problem: $\mathcal{S} \vdash_{\mathcal{J}} \rightarrow s$. Now to solve , $\mathcal{S} \vdash_{\mathcal{J}} S \rightarrow s$ observe that it is enough to erase the elements of S in all premises of clauses in $\mathcal{S}$ (yielding $\mathcal{S}'$) and check if $\mathcal{S}' \vdash_{\mathcal{J}} \rightarrow s$ which is decidable in PTIME.

Now we only have to use the previous oracle instead of the one of lemma 4.6 in the proof of lemma 4.7 yielding our theorem. □

4.3.1 Adding other predicate symbols

We now consider the case where the clauses c_s, c_n, c_c are guarded with literals built on a set of predicate symbols $\mathcal{P}$ not containing $\triangleright$ and that are defined using a saturated set of Horn clauses $\mathcal{A}_0$. For instance, $c_c(w)$ is extended to clauses of the form $P_1(s_1), \ldots, P_n(s_n),\ X \vdash y,\ X; w \vdash p \ \rightarrow X \vdash p$. The variables of $s_1, \ldots, s_n$ are assumed to be a subset of the variables of w, y.

We modify the rules Cut^i_w adding as premises the literals $P_1(s_1), ..., P_n(s_n)$. The precise formulation of these rules is presented in figure 4.4. For the rule cut_w we gather the predicates $P_1(s_1), ..., P_n(s_n)$ guarding the rule in the formula $\Psi_w(y, w)$. The mark inheritance is as in the previous subsection. We write $?^1 \wedge ?^2$ as syntactic sugar for the existence of a mark p if $?^1$ or $?^2$ is a mark.

Lemma 4.15 still holds, provided we add to $\mathcal{S}$ finitely many ground atoms on the new alphabet of predicates. To see this, we need to check that the proof transformation yields the same instances of $P_i(s_i)$. Lemma 4.16 is unchanged. These properties rely on the fact that guards (and their instances) do neither

depend on the set variable X (nor its instances) nor on the instances of the pattern.

Theorem 3 can then be extended to this case: when computing the fixed point, the instances of applicable inference rules are known at each step and we only have to check whether the corresponding instances of the guards are consequences of $\mathcal{A}_0$ (and possibly a finite set of ground atoms), which can be performed in PTIME, thanks to [BG01]. As a consequence, we get:

Theorem 4. *Let $\mathcal{P}$ be a set of predicate symbols, not containing $\triangleright, =$ and $\mathcal{A}_0$ be a set of Horn clauses built on $\mathcal{P}$ and which is saturated w.r.t. a basic ordering. If $\mathcal{S}$ is a set of ground clauses built on $\triangleright$ (possibly with guards using $\mathcal{P}$), we can decide in PTIME the satisfiability of $\mathcal{S} \cup \mathcal{A}_0$, together with T, W and finitely many clauses c_n, c_s, c_c, that are built on the same pattern p and which may be guarded by atomic formulae that use the predicate symbols in $\mathcal{P}$.*

$$\frac{\mathcal{A}_0 \vdash \Psi_w(x,w) \qquad A_1,\dots,A_n \to_? x \qquad B_1,\dots,B_m, w \to_? v}{A_1,\dots,A_n,B_1,\dots,B_m \to_p v}\ \mathsf{Cut}^1_w$$

$$\frac{\mathcal{A}_0 \vdash \Psi_w(x,w) \qquad A_1,\dots,A_n \to_? x \qquad w,B_1,\dots,B_m \to p}{A_1,\dots,A_n,B_1,\dots,B_m \to p}\ \mathsf{Cut}^2_w$$

$$\frac{A_1,\dots,A_n \to_{?^1} x \qquad B_1,\dots,B_m, x \to_{?^2} v}{A_1,\dots,A_n,B_1,\dots,B_m \to_{?^1 \wedge ?^2} v}\ \mathsf{Cut}^1$$

$$\frac{A_1,\dots,A_n \to_? x \qquad x,B_1,\dots,B_m \to p}{A_1,\dots,A_n,B_1,\dots,B_m \to p}\ \mathsf{Cut}^2$$

$$\frac{\mathcal{A}_0 \vdash \Psi_u(u) \quad A_1,\dots,A_n,u \to_? x}{A_1,\dots,A_n \to_p x}\ \mathsf{Str}^1_u$$

$$\frac{\mathcal{A}_0 \vdash \Psi_u(u) \quad A_1,\dots,A_n,u \to_? p}{A_1,\dots,A_n \to p}\ \mathsf{Str}^2_u$$

Figure 4.4: Inference rules for the case with additional predicates

4.3.2 Adding equality

We can extend again Theorem 4 to ground equalities in the atomic formulae of $\mathcal{S}$. The procedure is the same as in Lemma 4.7: for a fixed E, Lemmas 4.15 and 4.16 can be extended, considering representatives modulo $=_E$. Then we only have to compute a fixed point of a function f on the atomic formulae of $\mathcal{S}$, using the PTIME oracles provided by (extensions of) Lemmas 4.15 and 4.16. Note that in order to know if the Str_u rule of Cut_w rules can be applied, we need to decide the following problem: given a ground term t and a linear term u does it exist a σ such that $t =_E u\sigma$.

Lemma 4.17. *Given a ground term t, a ground equational theory E and a linear term u, checking the existence of σ such that $t =_E u\sigma$ in in* **PTIME**.

Proof. The proof is quite similar to the proof of lemma 4.9.

1. Build a tree automaton $\mathcal{A}$ (of polynomial size) that recognises the set of all v such that $v =_E t$.
2. Build a tree automaton $\mathcal{B}$ (of polynomial size in $|u|$) that recognises the language $\{u\sigma | \sigma : \mathrm{fv}(u) \to \mathcal{T}\}$.
3. Check (in polynomial time in $|\mathcal{A}| + |\mathcal{B}|$) whether $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$.

The only point that was not precised in the proof of lemma 4.9 is the second step. As u is a linear term, the problem is only recognising terms the top symbols of a term which should match the one of u. This can easily be done by a tree automaton of size linear in $|u|$. □

Given the previous lemma, obtaining the theorem is simply a matter of extending the inference rules as shown in figure 4.5 for them to work up-to the equational theory.

$$\frac{\mathcal{A}_0 \vdash \Psi_w(x,w) \quad A_1,\dots,A_n \to_? x \quad B_1,\dots,B_m, y \to_? v \quad \exists\sigma. y =_E w\sigma}{A_1,\dots,A_n,B_1,\dots,B_m \to_p v}\ \mathsf{Cut}^1_w(E)$$

$$\frac{\mathcal{A}_0 \vdash \Psi_w(x,w) \quad A_1,\dots,A_n \to_? x \quad w,B_1,\dots,B_m \to z \quad \exists\sigma. y =_E w\sigma,\ z =_E p\sigma}{A_1,\dots,A_n,B_1,\dots,B_m \to z}\ \mathsf{Cut}^2_w(E)$$

$$\frac{A_1,\dots,A_n \to_{?^1} x \qquad B_1,\dots,B_m, y \to_{?^2} v \qquad x =_E y}{A_1,\dots,A_n,B_1,\dots,B_m \to_{?^1\wedge ?^2} v}\ \mathsf{Cut}^1(E)$$

$$\frac{A_1,\dots,A_n \to_? x \qquad x,B_1,\dots,B_m \to z \qquad x =_E y, \exists\sigma. z =_E p\sigma}{A_1,\dots,A_n,B_1,\dots,B_m \to z}\ \mathsf{Cut}^2(E)$$

$$\frac{\mathcal{A}_0 \vdash \Psi_u(u) \quad A_1,\dots,A_n, y \to_? x \qquad \exists\sigma. y =_E u\sigma}{A_1,\dots,A_n \to_p x}\ \mathsf{Str}^1_u(E)$$

$$\frac{\mathcal{A}_0 \vdash \Psi_u(u) \quad A_1,\dots,A_n, u \to_? z \qquad \exists\sigma. y =_E u\sigma, z =_E p\sigma}{A_1,\dots,A_n \to z}\ \mathsf{Str}^2_u(E)$$

Figure 4.5: Inference rules for the case with additional predicates and equality

Theorem 5. *Let $\mathcal{P}$ be a set of predicate symbols, not containing $\triangleright, =$ and $\mathcal{A}_0$ be a set of Horn clauses built on $\mathcal{P}$ and which is saturated together with the theory of equality w.r.t. a basic ordering. If $\mathcal{S}$ is a set of ground horn clauses built on $\triangleright, =$ (possibly with guards using $\mathcal{P}$), we can decide in PTIME the satisfiability of $\mathcal{S} \cup \mathcal{A}_0$, together with T, W and finitely many clauses c_n, c_s, c_c, that are built on the same pattern p and which may be guarded by atomic formulae that use the predicate symbols in $\mathcal{P}$.*

4.4 The general case

Finally, we extend the results of the previous section to clauses with monotone constraints (note that we do not cover arbitrary constraints).

Adding a fixed set of possible constraints increases significantly the difficulty: Lemmas 4.15 and 4.16 no longer hold, as shown by the following example:

Example 4.18. Consider the clause $c_{f(y,k)} : \quad X \triangleright y,\ X; f(y,k) \triangleright n \ \rightarrow \ X \triangleright n \quad \| f(a,k), f(b,k), f(c,k) \notin X$. Consider the ground deducibility formulas: $\mathcal{S} = \{(f(a,k) \triangleright f(b,k),\ f(b,k); f(c,k) \triangleright n\}$. Does $c_{f(y,k)}$ and $\mathcal{S}$ entail $a; c \triangleright n$?

Following the procedure of Section 4.3,

$$\dfrac{\rightarrow c \qquad \dfrac{\dfrac{\rightarrow a \quad f(a,k) \rightarrow f(b,k)}{\rightarrow_p f(b,k)}\,\mathsf{Cut}^1_{f(y,k)} \qquad f(b,k); f(c,k) \rightarrow n}{f(c,k) \rightarrow n}\,\mathsf{Cut}^2}{\rightarrow n}\,\mathsf{Cut}^2_{f(y,k)}$$

in which each $\mathsf{Cut}^i_{f(y,k)}$ satisfies the constraint that $f(a,k), f(b,k), f(c,k)$ do not appear in the context: the instance of X is empty in each case. The procedure would then incorrectly answers "yes" to the entailment question.

Indeed, the proof rewriting of Lemma 4.15 yields the following (invalid) proof, in which the constraints are *not* satisfied in the first application of $\mathsf{Cut}_{f(x,k)}$, since the corresponding instance of X is the one element set $f(c,k)$:

$$\dfrac{\rightarrow c \qquad \dfrac{\rightarrow a \qquad \dfrac{f(a,k) \rightarrow f(b,k) \quad f(b,k); f(c,k) \rightarrow n}{f(a,k); f(c,k) \rightarrow n}\,Res}{f(c,k) \rightarrow n}\,\mathsf{Cut}_{f(x,k)}}{\rightarrow n}\,\mathsf{Cut}_{f(x,k)}$$

Our solution consists in designing another inference system, along the same ideas as before, for which Lemmas 4.15 and 4.16 still hold. To do so, we memorise more information in the mark (typically the constraints that need to be satisfied) so that the matching rule (removing the mark) can be applied only if the actual clauses would satisfy the constraints recorded in the mark.

Example 4.19. To explain the main idea, we give a simplified example of how the new proof system works. Coming back to Example 4.18, in our system we get:

$$\dfrac{\rightarrow a \quad f(a,k) \rightarrow f(b,k)}{\rightarrow_{f(a,b), f(b,k), f(c,k) \notin X} f(b,k)}\,\mathsf{Cut}^1_{f(y,k)}$$

But we cannot apply Cut^2 since its application requires that the context satisfies the constraint in the mark, which is not the case. We could apply a Cut^1, without removing the mark but then the mark could not be removed any more since the marks can never be removed from the "pattern premise" of a Cut^i_w rule.

If the clause is less constrained, for instance assume that we only impose

$f(b,k) \notin X$, then we can prove $\to n$ as follows:

$$\dfrac{\to c \qquad \dfrac{\dfrac{\to a \quad f(a,k) \to f(b,k)}{\to_{f(b,k)\notin X} f(b,k)}\,\mathsf{Cut}^1_{f(y,k)} \qquad f(b,k); f(c,k) \to n}{f(c,k) \to n}\,\mathsf{Cut}^2}{\to n}\,\mathsf{Cut}^2_{f(y,k)}$$

This time, we may remove the mark, as the instance of X is the singleton $\{f(c,k)\}$, that does not contain $f(b,k)$.

We get an analogue of Lemmas 4.15 and 4.16, which yields a PTIME decision procedure (because the number of possible marks is fixed), thus getting our main theorem.

Theorem 6. *If $\mathcal{S}$ is a set of ground clauses built on $\triangleright$, we can decide in PTIME the satisfiability of $\mathcal{S}$ together with T, W and finitely many constrained clauses c_s, c_c built on the same pattern p, provided the constraints are monotone.*

4.4.1 Proof of Theorem 6

Let us consider the constrained versions of the cuts and strengthening rules.

$$\textsc{Cut}_i \dfrac{X \to y \quad X', u_i(y) \to p}{X; X' \to p}\,\Gamma_i(X \cup X')$$

and

$$\textsc{Str}_i \dfrac{X; v_i \to p}{X \to p}\,\Delta_i(X)$$

where $\Gamma_i(X \cup X')$ (resp. $\Delta_i(X)$) guarding the rule $\textsc{Cut}_i$ (resp. $\textsc{Str}_i$) means that the rule may only be applied to $S \to t$ and $S'; u_i(t)\sigma \to p\sigma$ (resp. $S; v_i\sigma \to p\sigma$) if $S \cup S' \models \Gamma_i$ (resp. $S \models \Delta_i$).

In order to be able to have a unit proof strategy we need to be able to do some cuts early, even if the constraints are not satisfied. For that purpose, we introduce some constraint satisfaction obligations in the clauses: we consider now labelled clauses $S \ {}_{\Delta}\!\!\to_{\Gamma}\ t$ where Δ, Γ are finite sets of constraints. Intuitively the Δ mark records which constraints were satisfied at the introduction of the mark, while the Γ mark records which constraints will have to be satisfied when removing the mark.

Having introduced these new marks, let us generalise the $\textsc{Cut}_i$ and $\textsc{Str}_i$:

$$\textsc{Cut}^1_i \dfrac{X \ {}_{\Delta_1}\!\!\to_{\Gamma_1}\ y \quad X', u_i(y) \ {}_{\Delta_2}\!\!\to_{\Gamma_2}\ x}{X, X' \ {}_{\Delta_2}\!\!\to_{\Gamma_1,\Gamma_2,\Gamma_i}\ x}\,\Gamma_i(X \cup X'),\ \Gamma_i \in \Delta_1,\ \Gamma_1(X')$$

In the previous rule, $\Delta_1, \Delta_2, \Gamma_1, \Gamma_2$ are sets of constraints. By abuse of notation, we also view a set of constraint as representing the conjunction of all constraints in the set, thus giving a meaning to the condition $\Gamma_1(X')$. Intuitively, this rule allows delaying application of a $\textsc{Cut}_i$, checking that it could have been applied at the introduction of the mark ($\Gamma_i \in \Delta_1$), and its earlier application would not contradict other inferences made between the application of the mark and the

current step ($\Gamma_1(X')$). As in the previous section, we also need a way to remove marks:

$$\textsc{Cut}_i^2 \frac{X \ {}_{\Delta_1}\!\!\rightarrow_{\Gamma_1} y \quad X', u_i(y) \rightarrow p}{X, X' \rightarrow p} \Gamma_i(X \cup X'),\ \Gamma_i \in \Delta_1,\ \Gamma_1(X')$$

As previously, in this rule we check that all relevant conditions are satisfied. Moreover, the marks are removed because the clause ends with an instance of p, therefore there is no need to rewrite the proof to obtain a valid proof. Extending the $\textsc{Str}_i$ rule is simpler as we only need to record one constraint satisfaction obligation:

$$\textsc{Str}_i^1 \frac{X, v_i(y) \ {}_{\Delta}\!\!\rightarrow_{\Gamma} x}{X \ {}_{\Delta}\!\!\rightarrow_{\Gamma,\Delta_i} x} \Delta_i(X)$$

So far we have not defined any rule that introduce a mark, it is the scope of the next rule:

$$\textsc{Context} \frac{X \rightarrow x}{X \ {}_{\Delta} \rightarrow x} \Delta(X)$$

where $\Delta(X)$ will typically be the conjunction of all constraints satisfied by the instance of X in the rule application. This rule simply record that when we entered the marked zone, the constraint Δ was satisfied.

Our goal is now to prove that the new rules we added do not modify the entailment relation, as stated in the following lemma.

Lemma 4.20. *The previous* $\textsc{Cut}_i^1, \textsc{Cut}_i^2, \textsc{Str}_i^1, \textsc{Context}$ *are sound and complete with respect to* $\textsc{Cut}_i, \textsc{Str}_i$ *and weakening.*

Proof. Assume that $\mathcal{S}$ is a set of Horn clauses (without annotations with constraint sets) and that $S \rightarrow t$ is a clause, that is derivable in the inference system that includes the new extra rules. We show below that $S \rightarrow t$ is also provable without the extra rules.

We first note that, if one of the premises of a rule has a non-empty left or right constraint, it is also the case of the conclusion, except for the rule $\textsc{Cut}_i^2$. Therefore, any proof of a clause $S \rightarrow t$ that uses one of the additional rules, must also use at least once $\textsc{Cut}_i^2$. Consider a minimal (in size) proof Π of $S \rightarrow t$, that might use the extra rules. Consider a subproof Π' of Π that uses once $\textsc{Cut}_i^2$, as a last inference rule. We show that Π' can be rewritten into a strictly smaller proof (w.r.t. the size). This contradicts the minimality of Π, hence this proves that the minimal size proof does not make use of any extra rule.

First note that, according to labels inheritance, once a clause is annotated with sets of constraints, then the labels cannot be removed completely, unless we apply $\textsc{Cut}_i^2$. Since the leaves of Π' are not annotated with sets of constraints, we can write Π' as:

$$\dfrac{\dfrac{\vdots \quad \dfrac{\overset{\pi_1}{S^1 \to t}}{S^1 \ {}_{\Delta}{\to}_{\emptyset}\ t}\,\Delta_1^1(S_1)}{\vdots \qquad\quad \vdots}\,R^1}{S^n \ {}_{\Delta_1^n}{\to}_{\Gamma_1^n}\ t}\,R^n \qquad \overset{\pi_2}{S, u_i(t) \to p\sigma}}{S^n, S \to p\sigma}\,\Gamma_i(S^n, S), \Gamma_i \in \Delta_1^n, \Gamma_1^n(S)$$

where π_1, π_2 are proofs that do not use the extra rules and $R^1, \ldots, R^n$ are in $\textsc{Cut}_i^1, \textsc{Str}_i^1$. In particular, $\Gamma_1^1 \subseteq \ldots \subseteq \Gamma_1^n$ since these two rules only increase the right set of constraints.

We argue that Π' can be rewritten into

$$\dfrac{\dfrac{\vdots \quad \dfrac{\overset{\pi_1}{S^1 \to t} \qquad \overset{\pi_2}{S, u_i(t) \to p\sigma}}{S^1, S \to p\sigma}\,\Gamma_i(S^1, S)}{\vdots \qquad\quad \vdots}\,\widetilde{R^1}}{S^n, S \to p\sigma}\,\widetilde{R^n}$$

This is a strictly smaller proof, which is what we want. It only remains to define the rules $\widetilde{R^i}$ and check that the above proof is a valid proof in the new inference system indeed.

If

$$R^k = \dfrac{V_2^k \ {}_{\Delta_2^k}{\to}_{\Gamma_2^k}\ t^k \qquad V_1^k, u_i(t^k) \ {}_{\Delta_1^k}{\to}_{\Gamma_1^k}\ t}{S^k \ {}_{\Delta_1^k \cap \Delta_2^k}{\to}_{\Gamma_1^k, \Gamma_2^k, \Gamma_k}\ t}\Gamma_k(V_1^k, V_2^k),\ \Gamma_k \in \Delta_2^k,\ \Gamma_2^k(V_1^k)$$

we let

$$\widetilde{R_k} = \dfrac{V_2^k \ {}_{\Delta_2^k}{\to}_{\Gamma_2^k}\ t^k \qquad S, V_1^k, u_i(t^k) \to p\sigma}{S, S^k \to p\sigma}\Gamma_k(S, V_1^k, V_2^k),\ \Gamma_k \in \Delta_2^k,\ \Gamma_2^k(S, V_1^k)$$

The rule $\textsc{Cut}_i^1$ is therefore replaced with a rule $\textsc{Cut}_i^2$. The conditions are satisfied indeed (we get a valid proof):

- $\Gamma_k \in \Gamma_1^k \cup \Gamma_2^k \cup \{\Gamma_k\} = \Gamma_1^{k+1} \subseteq \Gamma_1^n$ and S satisfies Γ_1^n, hence $\Gamma_k(V_1^k, V_2^k) \to \Gamma_k(S, V_1^k, V_2^k)$
- $\Gamma_2^k \subseteq \Gamma_1^{k+1} \subseteq \Gamma_1^n$, and S satisfies Γ_1^n, hence $\Gamma_2^k(V_1^k) \to \Gamma_2^k(S, V_1^k)$

If

$$R^k = \dfrac{V_1^k, v_i(u^k) \ {}_{\Delta_1^k}{\to}_{\Gamma_1^k}\ t}{V_1^k \ {}_{\Delta_1^k}{\to}_{\Gamma_1^k \cup \{\Delta_i\}}\ t}\,\Delta_i(V_1^k)$$

we let

$$\widetilde{R^k} = \dfrac{S, V_1^k, v_i(u^k) \to t}{S, V_1^k \to t}\,\Delta_i(S, V_1^k)$$

The rule $\textsc{Str}_i^1$ is replaced with a rule $\textsc{Str}_i$. The condition is satisfied since , as before, $\Delta_i \in \Gamma_1^n$ and S satisfies Γ_1^n. □

The previous rules let us have a nicer proof shape, but they are still not sufficient to obtain completeness of a unit strategy. In order to reach this goal, we need to extend the rules even more. We add a new mark L that, intuitively, records a set of cuts that are done in advance. Note that remembering the precise cut is not useful, it is enough to remember the constraints involved in the cut. Effectively L is a multiset of facts $\neg\Gamma \to \neg\Delta$, recording the fact that as long as the constraint Γ is not satisfied, neither is the constraint Δ. The idea is that we do the cut in advance ignoring the fact that the constraint Γ is not yet satisfied, removing a literal that contradicts satisfaction of constraint Δ, but record that this cut is not effective (at least in terms of constraints) until the constraint Γ is satisfied. We start by extending the four previous rules with this new marking, keeping track of the fact that the constraints in L are not satisfied.

$$\textsc{Cut}_i^1 \frac{X\ \Delta_1 \to_{\Gamma_1}^{L_1} y \qquad X', u_i(y)\ \Delta_2 \to_{\Gamma_2}^{L_2} x}{X, X'\ \Delta_2 \to_{\Gamma_1,\Gamma_2,\Gamma_i}^{L_1,L_2} x}\ {\scriptstyle \Gamma_i(X\cup X'),\ \Gamma_i\in\Delta_1,\ \Gamma_1(X'),\ \forall(\neg\Gamma\to\neg\Delta)\in L_1,L_2.\Gamma_i,\Gamma_1\notin\Delta}$$

$$\textsc{Cut}_i^2 \frac{X\ \Delta_1 \to_{\Gamma_1}^{L_1} y \quad X', u_i(y) \to^{L_2} p}{X, X' \to^{L_1,L_2} p}\ {\scriptstyle \Gamma_i(X\cup X'),\ \Gamma_i\in\Delta_1,\Gamma_1(X'),\ \forall(\neg\Gamma\to\neg\Delta)\in L_1,L_2.\Gamma_i,\Gamma_1\notin\Delta}$$

$$\textsc{Str}_i^1 \frac{X, v_i(y)\ \Delta \to_{\Gamma}^{L_1} x}{X\ \Delta \to_{\Gamma,\Delta_i}^{L_1} x}\ \Delta_i(X),\ \forall(\neg\Gamma \to \neg\Delta') \in L_1.\Delta_i \notin \Delta'$$

$$\textsc{Context} \frac{X \to^{L} x}{X\ \Delta \to^{L} x}\Delta(X),\ \forall(\neg\Gamma \to \neg\Delta') \in L.\Delta' \notin \Delta$$

The four previous rules do not allow us to anticipate cuts (or strengthenings), this is the purpose of the next three rules. Note that we only give the unit version of these rules. The three next rules allow to perform a unit cut or a strengthening without checking that the associated constraint holds, and instead record that the constraints contradicted by the removed literal do not hold until the constraints necessary to actually apply the non-anticipated inference rule are satisfied.

$$\textsc{Cut}_i' \frac{\Delta_1 \to_{\Gamma_1}^{L_1} y \qquad X', u_i(y)\ \Delta_2 \to_{\Gamma_2}^{L_2} x}{X'\ \Delta_2 \to_{\Gamma_2}^{L_1,L_2,(\neg\Gamma_1,\Gamma_i\to\neg\Delta^c)} x}\Gamma_i \in \Delta_1,\ \Delta(u_i(y))$$

The intuitive meaning of the new element of L here is: as long as Γ_1, Γ_i are not satisfied (note that these are the constraints that would be checked when applying $\textsc{Cut}_i^1$), neither are the constraints contradicted by $u_i(y)$ (the complement of the constraints satisfied by $u_i(y)$). We proceed similarly for the two other rules.

$$\textsc{Cut}_i'' \frac{\Delta_1 \to_{\Gamma_1}^{L_1} y \quad X', u_i(y) \to^{L_2} p}{X' \to^{L_1,L_2,(\neg\Gamma_1,\Gamma_i\to\neg\Delta^c)} p}\ \Gamma_i \in \Delta_1,\ \Delta(u_i(y))$$

$$\textsc{Str}_i' \frac{X, v_i(y)\ \Delta \to_{\Gamma}^{L_1} x}{X\ \Delta \to_{\Gamma}^{L_1,(\neg\Delta_i\to\neg\Delta^c)} x}\ \Delta(u_i(y))$$

We are now left with the task of virtually applying the anticipated cut and strengthenings. This is in practice only removing an element of L. In order to

do so, one has to check that the corresponding constraints are satisfied, and add the corresponding constraint satisfaction obligations.

$$\textsc{Remove}_1 \frac{X \ {}_{\Delta_1}\!\!\rightarrow^{L,(\neg\Gamma\rightarrow\neg\Delta)}_{\Gamma_1} \ x}{X \ {}_{\Delta_1}\!\!\rightarrow^{L}_{\Gamma_1,\Gamma} \ x}\Gamma(X), \quad \forall(\neg\Gamma' \rightarrow \neg\Delta') \in L.\Gamma \notin \Delta$$

$$\textsc{Remove}_2 \frac{X \rightarrow^{L,(\neg\Gamma\rightarrow\neg\Delta)} p}{X \rightarrow^{L} p}\Gamma(X), \quad \forall(\neg\Gamma' \rightarrow \neg\Delta') \in L.\Gamma \notin \Delta$$

We now have to show that adding these new rules is sound with respect to the previous inference system.

Lemma 4.21. *If $\mathcal{S}$ entails a (possibly annotated) clause $S \ {}_{(\Delta)}\!\!\rightarrow_{(\Gamma)} \ t$ with the modified rules then $\mathcal{S}$ entails $S \ {}_{(\Delta)}\!\!\rightarrow_{(\Gamma)} \ t$ with rules $\textsc{Cut}_i, \textsc{Str}_i, \textsc{Cut}_i^1, \textsc{Cut}_i^2, \textsc{Str}_i^1$ and* Context*, cut and weakening.*

Proof. First of all, note that if a proof Π with the new rules does not use $\textsc{Cut}_i', \textsc{Cut}_i'', \textsc{Str}_i'$ then for all clauses $S \ {}_{(\Delta)}\!\!\rightarrow^{L}_{(\Gamma)} \ t$ in Π, L is empty. Note that with L empty, the old and the new version of $\textsc{Cut}_i^1, \textsc{Cut}_i^2, \textsc{Str}_i^1$ are the same as the old ones, therefore, Π is a valid proof in the old proof system.

Let Π be a proof of $S \ {}_{(\Delta)}\!\!\rightarrow_{(\Gamma)} \ t$. Assume that the number of rules $\textsc{Cut}_i', \textsc{Cut}_i'', \textsc{Str}_i'$ is minimal in Π. By contradiction assume that there is a rule $\textsc{Cut}_i', \textsc{Cut}_i'', \textsc{Str}_i'$ in Π. We will only treat the case of a $\textsc{Cut}_i'$ here, as this is the most involved case. Assume that it is the following $\textsc{Cut}_i'$ rule:

$$R^0 \frac{\Delta_1 \ {}\!\!\rightarrow^{L_1}_{\Gamma_1} \ u \qquad S, u_i(u) \ {}_{\Delta_2}\!\!\rightarrow^{L_2}_{\Gamma_2} \ v}{S \ {}_{\Delta_2}\!\!\rightarrow^{L_1,L_2,(\neg\Gamma_1,\Gamma_i\rightarrow\neg\Delta(u_i(u))^c)}_{\Gamma_2} \ v}\Gamma_i \in \Delta_1$$

As the conclusion of Π is not annotated by $(\neg\Gamma_1, \Gamma_i \rightarrow \neg\Delta(u_i(u))^c)$ there is in Π after the previous cut a Remove rule of the following form – assume that it is a $\textsc{Remove}_1$ rule (the $\textsc{Remove}_2$ case is similar) – we take R^n as the first occurrence of such a rule after R^0

$$R^n \frac{S' \ {}\!\!\rightarrow^{L,(\neg\Gamma_1,\Gamma_i\rightarrow\neg\Delta(u_i(u))^c)}_{\Gamma} \ v'}{S' \ {}_{\Delta}\!\!\rightarrow^{L}_{\Gamma,\Gamma_i,\Gamma_1} \ v'}\Gamma_i, \Gamma_1(S'), \quad \forall(\neg\Gamma' \rightarrow \neg\Delta') \in L.\Gamma_i, \Gamma_1 \notin \Delta'$$

Let $R^1, \ldots, R^{n-1}$ be the path in Π from the R^0 to R^n. If R^k is

$$R^k \frac{S_1^k \ {}_{\Delta_1}\!\!\rightarrow^{L_1^k}_{\Gamma_1} \ u^k \qquad S_2^k, u_i(u^k) \ {}_{\Delta_2}\!\!\rightarrow^{L_2^k}_{\Gamma_2} \ v^k}{S_1^k, S_2^k \ {}_{\Delta_2}\!\!\rightarrow^{L^k}_{\Gamma_1,\Gamma_2,\Gamma_i} \ v^k}{\scriptstyle \Gamma_i(S_1^k \cup S_2^k),\ {}'\Gamma_i \in \Delta_1,\ \Gamma_1(S_2^k),\ \forall(\neg\Gamma\rightarrow\neg\Delta)\in L_1,L_2.\Gamma_i,\Gamma_1\notin\Delta}$$

if R^{k-1} is the left premise of R^k, we take $\widetilde{R^k}$ as

$$\widetilde{R^k} \frac{S_1^k, u_i(u) \ {}_{\Delta_1}\!\!\rightarrow^{\widetilde{L^{k-1}}}_{\Gamma_1} \ u^k \qquad S_2^k, u_i(u^k) \ {}_{\Delta_2}\!\!\rightarrow^{L_2^k}_{\Gamma_2} \ v^k}{S_1^k, S_2^k \ {}_{\Delta_2}\!\!\rightarrow^{\widetilde{L^k}}_{\Gamma_1,\Gamma_2,\Gamma_i} \ v^k}{\scriptstyle \Gamma_i(S_1^k,S_2^k,u),\ \Gamma_i\in\Delta_1,\ \Gamma_1(S_2^k),\ \forall(\neg\Gamma\rightarrow\neg\Delta)\in L_1,L_2.\Gamma_i,\Gamma_1\notin\Delta}$$

With $\widetilde{L^k} = \widetilde{L^{k-1}}, L_2^k$. As $(\neg\Gamma_1, \Gamma_i \rightarrow \neg\Delta_{u_i}^c)$ is in L_2 (and $\Delta_{u_i}(u_i(u))$), we know that $\Gamma_i, \Gamma_1 \notin \Delta_{u_i}^c$, therefore, $\Gamma_i(u_i(u))$ holds and $\Gamma_1(u_i(u))$ holds, the constraints of $\widetilde{R^k}$ are satisfied.

We make a similar transformation if R^k is any other rule, and similar arguments ensures that that these transformations are correct. Note that $\widetilde{L^k} = L^k \backslash (\{(\neg\Gamma_1, \Gamma_i \to \neg\Delta_{u_i}^c)\} \cup L_1)$.

Now write:

$$\widetilde{R^n}\,\frac{\Delta_1 \to^{L_1}_{\Gamma_1} u \qquad S', u_i(u) \;\Delta_2 \to^{\widetilde{L^{n-1}}}_{\Gamma_2} v'}{S' \;\Delta_1 \cap \Delta_2 \to^{L_1, L_2}_{\Gamma_1, \Gamma_2, \Gamma_i} x}\;{\scriptstyle \Gamma_i(S'),\ \Gamma_i \in \Delta_1,\ \Gamma_1(S'),\ \forall (\neg\Gamma \to \neg\Delta) \in L.\Gamma_i, \Gamma_1 \notin \Delta}$$

Let Π' be Π in which we remove R^0 and for $k = 1..n$ we substitute $\widetilde{R^k}$ for R^k. The inference Π' is a valid inference of $S \;_{(\Delta)}\!\to_{(\Gamma)}\; t$, with one less $\textsc{Cut}'_i, \textsc{Cut}''_i, \textsc{Str}'_i$ rule than Π which contradicts our hypothesis. We conclude, that there is no $\textsc{Cut}'_i, \textsc{Cut}''_i, \textsc{Str}'_i$ in Π, therefore Π is an inference in the old inference system.

□

Having now proven that the our set of rules is indeed sound, we can finally reach the conclusion we wanted in the following lemma.

Lemma 4.22. *With the previous rules, a unit saturation strategy is complete.*

Proof. Let Π be a proof of $\to t$ with a minimal number of non unit rules. Let us assume, by contradiction that Π contains at least one non unit rule, let R^0 be a bottommost such rule, assume that R^0 is an instance of $\textsc{Cut}_i^2$

$$R^0\frac{S \;\Delta_1 \to^{L_1}_{\Gamma_1} u \quad S', u_i(u) \to^{L_2} p(v)}{S, S' \to^{L_1, L_2} p(v)}\;{\scriptstyle \Gamma_i(S \cup S'),\ \Gamma_i \in \Delta_1,\ \Gamma_1(S'),\ \forall (\neg\Gamma \to \neg\Delta) \in L_1, L_2.\Gamma_i, \Gamma_1 \notin \Delta}$$

We know that all rules following R^0 are unit rules, so there is a path in Π such that:

$$\cfrac{\cfrac{\cfrac{}{S^0 \to^{L_1, L_2} p(v)}\,R^0}{S^1 \to^{L^1} p(v)}\,R^1}{\vdots}$$
$$\cfrac{}{\to^{L^n} p(v)}\,R^n$$

Consider the sublist $(R^{k_1}, \ldots, R^{k_l})$ of $(R^i)_{i=1..n}$ where $S^{k_i} \backslash S^{k_i - 1} \in S$. Let us define $w^i = S^{k_i} \backslash S^{k_i - 1}$ and $\widetilde{S^i}$ as $S \backslash \{w_1, \ldots, w^i\}$.

Let us define inductively, for $i = 1..l$ the rules $\widetilde{R^{k_i}}$, $\widetilde{Rm^{k_i}}$ and $\widetilde{L^{k_i}}$. $\widetilde{L^0} = \emptyset$. If

$$R^{k_i}\frac{\Delta_1^i \to^{L_1^i}_{\Gamma_1^i} v^i \quad S^{k_i+1}, u_{j_i}(v^i) \to^{L^{k_i}} p(v)}{S^{k_i+1} \to^{L_1^i, L^{k_i}, (\neg\Gamma_1^i, \Gamma_{j_i} \to \neg\Delta(u_{j_i}(u^i))^c)} p(v)}\;\Gamma_{j_i} \in \Delta_1$$

then

$$\widetilde{Rm^{k_i}}\frac{\Delta_1^i \to^{L_1^i}_{\Gamma_1^i} v^i \quad \widetilde{S^i}, u_{j_i}(v^i) \to^{L_1, \widetilde{L^{k_i-1}}, \widetilde{L^{i-1}}} p(v)}{\widetilde{S^{i+1}} \to^{L_1^i, L_1, \widetilde{L^{k_i-1}}, (\neg\Gamma_1^i, \Gamma_{j_i} \to \neg\Delta(u_{j_i}(u^i))^c)} p(v)}\;\Gamma_{j_i} \in \Delta_1$$

and $\widetilde{L^{k_i}} = \widetilde{L^{k_i-1}}, (\neg\Gamma_1^i, \Gamma_{j_i} \to \neg\Delta(u_{j_i}(u^i))^c)$ and

$$\widetilde{R^{k_i}}\frac{S^{k_i} \backslash S \to^{L^{k_i-1}, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_i-1}})} p(v)}{S^{k_i+1} \backslash S \to^{L^{k_i}, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_i}})} p(v)}$$

Note that $\widetilde{R^{k_i}}$ is simply the identity rule. We define $\widetilde{Rm^{k_i}}, \widetilde{L^{k_i}}, \widetilde{R^{k_i}}$ the same way if R^{k_i} is an instance of $\textsc{Str}'_i$

If

$$R^{k_i}\frac{\Delta_1^i \xrightarrow[\Gamma_1^i]{L_1^i} v^i \quad S^{k_i+1}, u_{j_i}(v^i) \rightarrow^{L^{k_i}} p(v)}{S^{k_i+1} \rightarrow^{L_1^i, L^{k_i}} p(v)} \begin{array}{c}\Gamma_{j_i} \in \Delta_1^i, \Gamma_1^i(S^{k_i+1}), \Gamma_1^i(S^{k_i+1}),\\ \forall(\neg\Gamma \rightarrow \neg\Delta) \in L_1^i, L^{k_i}.\Gamma_{j_i}, \Gamma_1^i \notin \Delta\end{array}$$

then

$$\widetilde{Rm^{k_i}}\frac{\Delta_1^i \xrightarrow[\Gamma_1^i]{L_1^i} v^i \quad \widetilde{S^i}, u_{j_i}(v^i) \rightarrow^{L_1, \widetilde{L^{k_{i-1}}}, \widetilde{L^{i-1}}} p(v)}{\widetilde{S^{i+1}} \rightarrow^{L_1^i, L_1, \widetilde{L^{k_{i-1}}}, (\neg\Gamma_1^i, \Gamma_{j_i} \rightarrow \neg\Delta(u_{j_i}(u^i))^c)} p(v)} \Gamma_{j_i} \in \Delta_1$$

and $\widetilde{L^{k_i}} = \widetilde{L^{k_{i-1}}}, (\neg\Gamma_1^i, \Gamma_{j_i} \rightarrow \neg\Delta(u_{j_i}(u^i))^c)$ and

$$\widetilde{R^{k_i}}\frac{S^{k_i} \backslash S \rightarrow^{L^{k_i-1}, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_{i-1}}})} p(v)}{S^{k_i+1} \backslash S \rightarrow^{L^{k_i}, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_i}})} p(v)} \begin{array}{c}\Gamma_{j_i}(S^{k_i} \backslash S), \Gamma_1^i((S^{k_i} \backslash S),\\ \forall(\neg\Gamma \rightarrow \neg\Delta) \in L^{k_i}, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_i}}).\Gamma_{j_i}, \Gamma_1^i \notin \Delta\end{array}$$

Note that $\widetilde{R^{k_i}}$ is a valid Remove rule. We define $\widetilde{Rm^{k_i}}, \widetilde{L^{k_i}}, \widetilde{R^{k_i}}$ the same way if R^{k_i} is an instance of $\textsc{Str}_i^1$.

If $j \notin \{k_1, \ldots, k_l\}$, if $k_i < j < k_{i+1}$ we define $\widetilde{R^j}$ as

$$\widetilde{R^j}\frac{S^{j-1} \backslash S \rightarrow^{L^{j-1}, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_i}})} p(v)}{S^j \backslash S \rightarrow^{L^j, (\widetilde{L^{k_l}} \backslash \widetilde{L^{k_i}})} p(v)} R^j$$

Now note that

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{S \xrightarrow[\Gamma_1]{\Delta_1} u}{\vdots}\widetilde{R}m^{k_1}}{\Delta_1 \xrightarrow[\Gamma_1]{L_1, \widetilde{L^{k_l}}} \widetilde{R}m^{k_l} \qquad S', u_i(u) \rightarrow^{L_2} p(v)}\widetilde{R^0}}{S^0 \backslash S \rightarrow^{L^0, \widetilde{L^{k_l}}} p(v)}\widetilde{R^1}}{S^1 \backslash S \rightarrow^{L^{1'}} p(v)}}{\vdots}\widetilde{R^n}}{\rightarrow^{L^n} p(v)}$$

is a valid proof of $\rightarrow^{L^n} p(v)$ with one less non unit cut, which contradict our hypothesis.

□

Let us call $\mathcal{K}$ the previous set of inference rules. Having proven that a unit strategy is complete with respect to $\mathcal{K}$ it is only a formality to prove that as in the previous section, the entailment problem for atoms is decidable in **PTIME**.

Lemma 4.23. *The problem* $S_1 \rightarrow t_1, \ldots, S_n \rightarrow t_n \vdash_{\mathcal{K}} S \rightarrow t$ *is in* **PTIME**

Proof. In $S \ {}_{\Delta}{\to}^{L}_{\Gamma} \ t$ it is easy to see that whether the multiplicity of $x \in L$ is 2 or strictly greater than 2 is not relevant, as if x appears 2 times in L and a REMOVE can be applied for x (yielding a clause C annotated with L' with x of multiplicity 1 in L'), then it can be applied repeatedly if x appears more than twice in order to yield the same C.

First of all note that we know how to decide the problem $S_1 \to t_1, \ldots, S_n \to t_n \vdash_{\mathcal{K}} \to t$ as we can apply a unit strategy, and there are only a bounded number of annotations the decision procedure is in PTIME.

Now in order to decide the entailment problem, let Λ be a new constraint such that for all S, $\neg\Lambda(S)$. Now for every clause in $C_i = S_i \to t_i$ and every constraint Γ let $S_i^{\Gamma} = S_i \backslash \{u \in S | \neg\Gamma(u)\}$ and $C_i^{\Gamma} = S_i^{\Gamma} \to^{(\neg\Lambda \to \neg\Gamma)}$. We observe that $S_1 \to t_1, \ldots, S_n \to t_n \vdash_{\mathcal{K}} S \to t$ iff there exists Γ such that $S_1 \to t_1, \ldots, S_n \to t_n \vdash_{\mathcal{K}} \to^{(\neg\Lambda \to \neg\Gamma)} t$. Clearly if $S_1 \to t_1, \ldots, S_n \to t_n \vdash_{\mathcal{K}} \to^{(\neg\Lambda \to \neg\Gamma)} t$ (i.e. Π be a proof of $\to^{(\neg\Lambda \to \neg\Gamma)} t$ in $\mathcal{K}$) then as the $(\neg\Lambda \to \neg\Gamma')$ annotations can never be removed, replacing the leafs of Π that are C_i^{Γ} by $C_i = C_i^{\Gamma} \cup \{u \in S | \neg\Gamma(u)\}$ yields a proof of $S' \to t$ with $S' \subseteq S$. Conversely, if there exists a proof of $S' \to t$ with $S' \subseteq S$ with $S' \subseteq S$ without weakening, then one can build a proof of $\to^{(\neg\Lambda \to \neg\Gamma)} t$ from the C_i^{Γ} by backtracking the origin of the atoms in S'. □

From the **PTIME** decidability of the entailment problem from atoms as given in the previous lemma, we can (easily) prove our main theorem with the same proof technique as for the simpler cases.

proof of Theorem 6. Computing the least fixed point of the function defined in the proof of lemma 4.7 using the oracle computing whether $S_1 \triangleright t_1, \ldots, S_n \triangleright t_n \vdash_{\mathcal{K}} S \triangleright t$ yields a PTIME decision procedure. □

4.5 Conclusion

Beyond our tractability results, our techniques and ideas are reused in SCARY in order to design an efficient strategy. The most crucial idea that is reused is transforming axioms into inference rules. Moreover, the techniques of computing these inference rules modulo equality caries over in SCARY. During early experiments with SCARY, we discovered that non-monotone constraints were necessary to prove some protocols. We thus lose the **PTIME** complexity. However, we hope that the memorisation strategies we devised here may indeed be reused in other contexts.

Chapter 5

SCARY

In this chapter we describe the Scary (Security: Computational Attacks Reliable Yielder) tool based on the ideas of the previous chapter. The main part of the tool consists of an implementation of (a variant) of the decision procedure described in the previous chapter. The main difference is that we drop the **PTIME** requirement because of the complexity of keeping track of the constraints used and therefore move to an NP decision procedure. We will start by describing the computational axioms and the signature we consider. We will then describe the actual behaviour of the tool, starting with how we perform the necessary operations on terms such as matching modulo an equational theory or computing constraints, and then describe the global saturation procedure.

The aim of the tool is to decide the following problem:

Input A finite protocol P using only the functions described in the next section and equality tests

Output If there exists an attack, the trace on which the attack is possible together with a model of the attack. Secure otherwise.

The tool works in two phases: first compute a symbolic representation of all the traces of P, and the corresponding sets of formulae, second check, for each set of formulae, if it is satisfiable together with the axioms described in the next section. Attack finding is in overall in NP (including trace guessing).

We will only describe very shortly the first phase as it is very straightforward. However we will describe in details the second part of the tool, that take a set of ground equalities and $\triangleright$ atoms and returns a model if there exists one or unsatisfiable otherwise.

5.1 Overview of the tool

Let us first present a brief overview of the inner working of the tool. The first remark is that the decision procedure described in chapter 4 take as input a set of clauses where our tool take as input a protocol specified in a small concurrent calculus. Therefore, the first task of the tool is to compute all traces of the protocol and compute the sets of clauses corresponding to each trace together with the security property. We will not detail the algorithm used for unravelling the traces of the protocol, however we explain the generation of the set of formulae corresponding to a trace. This first part is done by

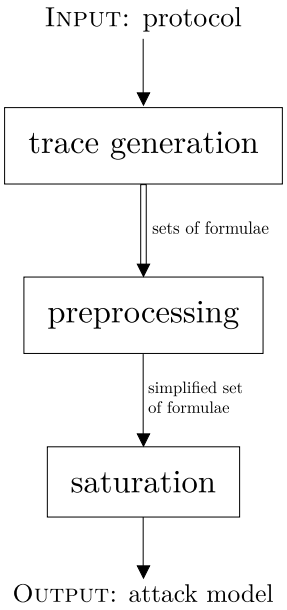

Figure 5.1: SCARY schematics

an independent module so it would be fairly easy to support various process algebras.

Then the main task is checking the satisfiability of each set of clauses corresponding to a trace together with the axioms we consider. However, as could be seen in the previous chapter, dealing with the functionality axiom can be cumbersome, therefore a small preprocessing phase is necessary, in order to generate all potentially useful instances of the functionality axiom. Another important part of the preprocessing phase is processing the equational theory in order to transform it into a convergent rewriting system. We also use this phase to perform some optimisations, breaking pairs that appear at top-level in the $\triangleright$ atoms, and removing some trivially deducible terms from the left hand side of $\triangleright$ atoms.

The main part of the tool is the saturation procedure. The first difficulty is the computations that need to be performed on terms. Namely, computing a convergent rewriting system from the equational theory, filtering modulo this equational theory and computing constraints. A big part of the difficulty of computing constraints comes from the fact that we want to know if, for a set of terms S, there exists a set of terms $S' =_E S$ that satisfies the constraint, instead of just checking whether S satisfies the constraint. Apart from these problems the other difficulty is the unusual branching on equality when applying integrity axioms.

5.2 Input syntax and trace computation

Let us start by defining the process algebra used in our input syntax. We use here a very simple process algebra, quite close to the basic processes defined

in [CC08]. A process in this algebra is a finite parallel composition of sequential processes without else branches. All nonces and secret keys are assumed to be restricted at top-level, all names and public keys are assumed to be public. The semantics of this process algebra is quite straightforward, there is only one input channel, one output channel and all communications are scheduled by the adversary. Let us give a more precise definition of this process algebra. First the sequential processes are defined by the following grammar:

$$\begin{array}{rcll} P & = & \mathbf{0} & \text{null process} \\ & | & in(x).P & \text{input} \\ & | & out(t).P & \text{output} \\ & | & [\Psi]P & \text{conditional } \texttt{if } \Phi \texttt{ then P else } \mathbf{0} \end{array}$$

where terms are built over the functions symmetric and asymmetric encryption, pairing and projections and the constants nonces, names and keys (symmetric and asymmetric). The condition Ψ is assumed to be a conjunction of equalities between terms. Now the basic processes we consider are parallel compositions of these sequential processes $B = P \| \ldots \| P$. Note that while this process algebra is quite minimalist, it is large enough to encode most cryptographic protocols as else branches are mostly used in processes that should have indistinguishability properties. Anyway, it is, in principle, quite easy to extend to procedure to processes involving non-trivial branching. The only unusual point is that we forbid replication of a process. However, as precised in chapter 2, this model is only sound for finite processes, moreover we need to make a satisfiability check for each trace, therefore we need to ensure that there are finitely many traces of a protocol. The only security property we consider is secrecy. More precisely, we consider security properties of the form $\neg\phi \triangleright s$ where s may be a nonce or a secret key and ϕ is the frame corresponding to the trace. Again, extensions to other reachability properties is just a matter of time; there is no obstacle in principle.

In the input syntax of our tool we add some syntactic sugar allowing us to define roles, and substitute terms for variables in the protocol. We will also implicitly assume that all randomnesses used for encryptions are valid randomnesses (according to definition 3.14). This is not a strong restriction as it only means that the randomnesses of encryptions are only used once, which is always the case in practice. Let us give a short example of this input syntax, it is an encoding of the NSL protocol with one honest session and one session of the responder communicating with the adversary.

```
role(Initiator):
  out(aenc(pair(a,na),pkb,ra1)).in(a1).
  [pi1(pi1(adec(a1,ska))) = b ; pi2(pi1(adec(a1,ska))) = na].
  out(aenc(pi2(adec(a1,ska)),pkb,ra2));
role(Responder):
  in(b2).[pi1(adec(b2,skb)) = a].
  out(aenc(pair(pair(b,pi2(adec(b2,skb))),nb),pka,rb1));

role(Initiator)
  (
    a -> name(a),
    na -> nonce(a),
```

```
    pkb -> pk(2),
    ska -> sk(1),
    ra1 -> nonce(ra1),
    ra2 -> nonce(ra2)
  )
|| role(Responder)
  (
    skb -> sk(2),
    a -> name(a),
    nb -> nonce(b),
    pka -> pk(1),
    rb1 -> nonce(rb1)
  )
|| role(Responder)
  (
    skb -> sk(2),
    a -> name(i),
    nb -> nonce(bi),
    pka -> pk(3),
    rb1 -> nonce(rb1i)
  )

honest_key(1), honest_key(2) => secret(nonce(b));;
```

The first part defines prototypes for the roles of initiator and responder, the second part instantiates these roles with the correct public keys and nonces, one initiator A talking to B, and two responders B: one talking to A for the honest session and one talking to I the attacker. The last part encodes the security property. For simplicity reasons we classify keys between honest keys (that have been honestly generated) and dishonest keys (that are adversarial keys). In practice these honest keys will be treated as constants of the sort key while dishonest keys will be considered as another sort that does not necessarily comply to the cryptographic axioms but complies with the equational theory of encryption. The last line of the input example simply states that if we assume that the keys 1 and 2 are honest then nonce nb should be secret.

We will not go into the details of how we generate all traces as it is pretty straightforward. Note that considering this process algebra, it is very easy to compute all possible interleavings of actions. Now translating these interleaving of actions into the model of chapter 2 is immediate. We are left with generating the clause sets corresponding to each trace. This is done as one could expect. The only somewhat non-trivial part is the security property. Let us consider a trace t, it is easy to compute the frame ϕ corresponding to this trace, we therefore add to the clauses set S_t corresponding to this trace the atom $\phi \triangleright s$ which is the negation of the security property. We also specify in the clause set which keys are honest. Handles are functions of the frame (as it is when they are generated). All instances of the application of one particular handle function are, however, equal. This allows to see handles as constants as the arguments of the handles are irrelevant as far as equalities are concerned. Therefore, we code handles as constants, and store the arguments of each handle using the `handle_dependency` predicate. At this point our tool generates a file coding for

the clause set. Let us give a short example of such a file:

```
honest_key(2);
handle_dependency(handle(1), [aenc(nonce(2),pk(2),nonce(123))]);
(aenc(nonce(2),pk(2),nonce(123))) |- handle(1);
pi2(adec(handle(1),sk(2))) = name(3) ;
(
  aenc(nonce(2),pk(2),nonce(123)) ;
  aenc(pair(pair(pi1(adec(handle(1),sk(2))),nonce(3)),name(2)),pk(3),nonce(124))
) |- nonce(2);;
```

For simplicity reasons we do not give the full example of a trace of NSL. The trace specified here corresponds to a trace where one honest session has been completed. We dropped all messages from this honest session except the last one. We want to prove secrecy of the nonce involved in this last message. More precisely the trace is:

$$\begin{array}{lcl} A & \rightarrow & \{n_b\}_{pk_b} \\ B & \leftarrow & \{x_n, I\}_{pk_b} \\ B & \rightarrow & \{x_n, n', B\}_{pk_I} \end{array}$$

we wish to check the secrecy of n_b. Note that this is only a session of the responder talking with an adversary that has some knowledge from a previous session.

5.3 The logic

We mainly rely on axioms presented in chapter 3, with several variations that are explained here.

We fix once and for all the signature of the logic as follows:

- $\mathcal{F} = \{\text{senc}/3, \text{aenc}/3, \text{sdec}/2, \text{adec}/2, \text{pair}/2, \pi_1/1, \pi_2/1, \text{handle}_i/i\}$
- $\mathcal{C} = \{\text{key}(\mathsf{int}), \text{pk}(\mathsf{int}), \text{sk}(\mathsf{int}), \text{nonce}(\mathsf{int}), \text{name}(\mathsf{int}), \text{var}(\mathsf{int})\}$
- $\mathcal{P} = \{=, \text{wf}\}$

Note that this restricts the class of protocols we study to be built only on encryptions, decryptions and pairing. It would not be difficult, however, to extend the tool to a larger set of functions or constants. Note that the handles are functions, but as precised in the previous section, they will be considered as constants when computing equalities. We will call the arguments of one particular handle function the dependencies of the handle.

Let us now describe the set of axioms $\mathcal{A}$ we consider. First of all we have axioms for the equational theory of encryption and pairing, let us call this equational theory $E_{\mathcal{F}}$:

$$\begin{array}{rcl} \pi_1(\text{pair}(x, y)) & = & x \\ \pi_2(\text{pair}(x, y)) & = & y \\ \text{sdec}(\text{senc}(x, \text{key}(i), y), \text{key}(i)) & = & x \\ \text{adec}(\text{aenc}(x, \text{pk}(i), y), \text{sk}(i)) & = & y \end{array}$$

We obviously consider the reflexivity, transitivity, monotonicity and functionality axioms. As we do want to derive contradictions, we need the freshness axiom:

$$\neg X \triangleright x \quad \| \, \text{fresh}(x, X)$$

We also consider the CCA versions of the secrecy and non-malleability axioms for asymmetric encryptions. We assume therefore that our cryptographic scheme is IND-CCA and which key concealing.

$$X, \{x\}^r_{\mathrm{pk}(i)} \triangleright \mathrm{nonce}(j) \;\rightarrow\; X \triangleright \mathrm{nonce}(j) \parallel \mathrm{usable}^{\mathrm{CCA}}(\mathrm{sk}(i), (X, x))$$

$$X \triangleright y \quad X, \mathrm{adec}(y, \mathrm{sk}(i)) \triangleright \mathrm{nonce}(i) \quad \rightarrow \bigvee_{\{x\}^r_{\mathrm{pk}(i)} \in st(X)} y = \{x\}^r_{\mathrm{pk}(i)} \parallel C$$

with C the conjunction of the following constraints:

- $\mathrm{usable}^{\mathrm{CCA}}(\mathrm{sk}(i), X)$
- $\mathrm{nonce}(i)$ only appears under encryption by X usable keys

Note that the non-malleability axiom does not comply to the form of axioms considered in chapter 4. In the decision procedure we did not consider the branching on the equality with honest encryptions. We believe however (and it seems to be confirmed by experimental results) that this branching does not harm the computation time too much as the number of honest encryptions with one given key is usually much lower than the number of terms.

For symmetric encryption we use the secrecy and the integrity axioms, which means that one should make sure that the encryption scheme used is IND-CPA, INT-CTXT and which-key concealing. The secrecy axiom is as before:

$$X, \{x\}^r_{\mathrm{key}(i)} \triangleright \mathrm{nonce}(j) \quad \rightarrow \quad X \triangleright \mathrm{nonce}(j) \parallel \mathrm{usable}^{\mathrm{CCA}}(\mathrm{key}(i), (X, x))$$

We use a weaker form of the integrity axiom which is a logical consequence of the axioms of chapter 3. Recall the original integrity axiom:

$$X \triangleright y \quad \mathrm{wf}(\mathrm{sdec}(y, \mathrm{key}(i))) \quad \rightarrow \bigvee_{\{z\}^r_{\mathrm{key}(i)} \in \mathrm{st}(X)} y = \{z\}^r_{\mathrm{key}(i)} \quad \parallel \mathrm{usable}^{\mathrm{CCA}}(\mathrm{key}(i), X)$$

Now, let C be a context such that if $\bot$ is the public failure constant, then computationally $C[\bot]$ evaluates to $\bot$ (i.e. $\neg(\mathrm{wf}(x)) \rightarrow C[x] = \bot$ is sound). Then the following axiom is a logical consequence of the integrity axiom:

$$X \triangleright y \quad Y, C[y] \triangleright z \quad \rightarrow$$
$$Y, \bot \triangleright z \quad \vee \bigvee_{\{z\}^r_{\mathrm{key}(i)} \in \mathrm{st}(X)} y = \{z\}^r_{\mathrm{key}(i)} \quad \parallel \mathrm{usable}^{\mathrm{CCA}}(\mathrm{key}(i), X)$$

Now note that as $\bot$ is a public constant, then $Y, \bot \triangleright z \leftrightarrow Y \triangleright z$ is computationally sound, therefore we can use the following integrity axiom:

$$X \triangleright y \quad Y, C[y] \triangleright z \quad \rightarrow$$
$$Y \triangleright z \quad \vee \bigvee_{\{z\}^r_{\mathrm{key}(i)} \in \mathrm{st}(X)} y = \{z\}^r_{\mathrm{key}(i)} \quad \parallel \mathrm{usable}^{\mathrm{CCA}}(\mathrm{key}(i), X)$$

Note that we use this axiom for a fixed list of possibly useful contexts satisfying the condition. That allows us to prove most protocols and can be easily extended if needed. As we do not want to fix any axiom for wf, the tool does not check whether $\neg\,\mathrm{wf}(x) \rightarrow \neg\,\mathrm{wf}(C[x])$. In practice it means that if one wants to extend this list of contexts, he should careful to check that this property is satisfied.

5.4 Computing on terms

The decision procedure relies on a few algorithms on terms, that we describe first. They solve the following problems:

1. Given a set of ground equations E, and two ground terms u, v, do we have $u =_{E \cup E_{\mathcal{F}}} v$?
2. Given a ground term u and a term $v(\overline{x})$, is there a substitution σ such that $u =_{E \cup E_{\mathcal{F}}} v\sigma$ with σ satisfying some constraints?
3. Decide the constraints from chapter 3

Both computing constraints and deciding equational theories are necessary for deciding the axioms applications. We have to be careful: we are computing modulo $=_{E \cup E_{\mathcal{F}}}$.

Note that instead of dealing only with a ground equational theory, we add equations for encryptions and pairs. While this is not strictly necessary as we could only add the necessary ground instances of this equational theory (as mentioned in the previous chapter), it simplifies the usage of this tool. In particular we do not have to guess which equations are necessary to prove security and which are not.

5.4.1 Equational theory

In order to decide these three problems, we transform the equational theory into a flat convergent rewriting system. It will be made clear later how the flatness condition simplifies both the unification problem and the computation of constraints satisfaction.

In order to build a flat convergent rewriting system, let us introduce new constants $\mathcal{C}_0 = \{\text{const}(i) | i \in \mathbb{N}\}$ to our alphabet. These constants "do not count" for the solutions of constraints. More precisely, we compute constraints up to the equational theory, and when doing so we disregard any representative of a term involving one of these new constants.

The first phase is flattening equations. More precisely, for any subterm t in E we add an equation $t = c_t$ for a fresh $c_t \in \mathcal{C}_0$. That way we may assume that all equations in E have one of the two following forms: either $f(a_1, \ldots, a_n) = b$ or $a_1 = b$ with $f \in \mathcal{F}$ and $a_1, \ldots, a_n, b \in \mathcal{C} \cup \mathcal{C}_0$.

We may now complete the system of equations with the following algorithm. We start by orienting all equations according to a lexicographic path ordering with $\mathcal{F} > \mathcal{C}_0 > \mathcal{C}$ thus obtaining a rewriting system that we now need to complete, together with $E_{\mathcal{F}}$. First of all note that (modulo equalities between constants) any superposition of a ground flat equation with an equation in $E_{\mathcal{F}}$ yields a flat ground equation. The completion algorithm computes equivalence classes of constants as follows: as long as a fix point is not reached,

- replace every constant in E with the smallest constant in its equivalence class
- lexicographically sort the rules that have a non-constant left hand side
- for each pair of rules with the same left hand side, either remove one of the two if their right hand side are equal, otherwise merge the classes of their right hand side.

This algorithm is correct as it is simply a $O(n^2 \log n)$ implementation of a Knuth-Bendix completion.

We now explain how we compute matching modulo an equational theory. We solve the following problem: given a ground term u and a term $v(\overline{x})$, is there a substitution σ such that $u =_{E \cup E_{\mathcal{F}}} v\sigma$ with σ satisfying some constraint C. We make the assumption that v is linear. This assumption could easily be dropped at the cost of an increased complexity of the procedure. However, as all instances we have to consider in our tool are linear, this restriction has not been a limitation so far.

Input: E (ground, flat, convergent), t linear term without destructors (i.e. t in $T(\mathcal{F} \setminus \{\mathrm{adec}, \mathrm{sdec}, \pi_1, \pi_2\}, \{x_1, \ldots, x_n\})$), C a constraint, u a ground term

Output: Is there a ground substitution σ such that we have both $t\sigma =_{E \cup E_{\mathcal{F}}} u$ and $x_i\sigma \models C$ for each x_i in the domain of σ?

We solve this problem using the following algorithm: start by normalising t and u (it preserves the existence of a solution)

1. If t is ground, check $t = u$
2. If t is a variable, check $u \models C$
3. If $t = f(t_1, \ldots, t_n), u = f(u_1, \ldots, u_n)$, recursively call the algorithm on $(t_1, u_1), \ldots (t_n, u_n)$. If all calls succeeds then output true. This step is correct as the term is linear, thus all calls are independent.
4. If $t = f(t_1, \ldots, t_n)$, and $u = g(u_1, \ldots, u_m)$ with $f \neq g$, then u must be a constant $\mathcal{C}$ (otherwise output false). In that case: for each rule $f(g_1, \ldots, g_n) \to u$, recursively call the algorithm on $(t_1, g_1), \ldots, (t_n, g_n)$. Output true is for one of the rules all the calls succeed:

$$\mathsf{match}(t, u) = \bigvee_{f(g_1,\ldots,g_n) \to u} \bigwedge_{i=1}^{n} \mathsf{match}(t_i, g_i)$$

Proof. The two base cases, cases 1 and 2 are clearly correct. Let us consider case 3. If $t = f(t_1, \ldots, t_n)$ and $u = f(t_1, \ldots, t_n)$, assume $t\sigma =_E u$. As u is irreducible, we have $t\sigma \to_E^* u$. As t does not contain the $\pi_i, \mathrm{sdec}, \mathrm{adec}$ symbols, the equations of $E_{\mathcal{F}}$ may not be applied to the head of the term. As the rewriting system for E is flat, there is no rule of the form $f(\_, \ldots, \_) \to u$. We conclude that $t_i\sigma = u_i$ for $i = 1, \ldots, n$.

In case 4, if u is not a constant, the considerations for the proof of case 3 ensure that there is no solution. If u is a constant and there exists a solution, then there exists σ such that $t\sigma \to_E^* u$. We therefore have one rewriting rule in E such that $f(v_1, \ldots, v_n) \to_E u$ for some $v_i =_E t_i\sigma$. We conclude that one of the calls succeeds. □

5.4.2 Constraints

We explain here how the satisfaction of the constraints involved in the axioms of chapter 3 is computed. For a formal definition of these constraints see the aforementioned chapter.

Answering the question "does t satisfy C?" for a term t and a constraint C is quite simple. However, we have to answer the more complicated question "is there a $u =_E t$ such that u satisfies C?" Solving this question involves searching antecedents modulo the equational theory. Note that the constants const(_) we

added at the flattening step are not part of the logic, therefore the constraints are not defined on terms containing such constants. We then need to search for an antecedent modulo E which does not involve constants in $\mathcal{C}_0$.

In the following section, given an equational theory E, as a shortcut we will say that $S_1, \ldots, S_n \models C(X_1, \ldots, X_n)$ if there exists $S'_1 =_{E \cup E_{\mathcal{F}}} S_1, \ldots S'_n =_{E \cup E_{\mathcal{F}}}$ such that $S'_1, \ldots, S'_n \models C(X_1, \ldots, X_n)$.

Freshness

The freshness constraint is the simplest as there is no case disjunction on the form of the term. Let us, in the following, assume that S is a set of ground terms in normal form.

First of all, let us remark that $S \models \text{fresh}(\text{nonce}(i), X)$ if for all t in S, we have $t \models \text{fresh}(\text{nonce}(i), X)$. Therefore solving the problem for one term is enough. It is now sufficient to remark that $\text{nonce}(i)$ is fresh in t if one of the following conditions is satisfied:

1. $t \in \mathcal{C}$ with $t \neq \text{nonce}(i)$ and t is not a handle
2. $t = h$ with h handle and, if D are the dependencies of h, $\text{nonce}(i)$ is fresh in D
3. $t = f(t_1, \ldots, t_n)$ and $\text{nonce}(i)$ is fresh in $t_1, \ldots, t_n$
4. $t \in \mathcal{C} \cup \mathcal{C}_0$ and there is a rule $f(t_1, \ldots, t_n) \to t$ in E with $\text{nonce}(i)$ fresh in $t_1, \ldots, t_n$ (if the first case does not apply).

Note that in order to force termination of this algorithm it is enough to ensure that the same rewriting rule may not be applied indefinitely in case 4. In order to achieve this, it is enough to make sure that we do not apply the same rule twice when we start looking at antecedents with the equational theory. Indeed, if a rewrite rule $f(t_1, \ldots, t_n) \to t$ is applied once in case 4 then we have to prove freshness of $\text{nonce}(i)$ in $t_1, \ldots, t_n$. Applying this case again with the same rewrite rule would leave us with the task of proving that $\text{nonce}(i)$ in $t_1, \ldots, t_n$ again.

The only thing to prove for the correction of this algorithm is the fact that we do not need to apply $E_{\mathcal{F}}$. This is clear as if n is not fresh in t, it is not fresh either in $\pi_1(\text{pair}(t, x))$. The same argument is valid for the equations of encryption.

Key protection

We show here how to decide the key-usability constraint. As in chapter 3, we start by computing key protection.

We will denote by $\text{maxkeyprotect}(t, K)$ one maximal set of keys protected by the set of keys K. More precisely, $\text{maxkeyprotect}(t, K)$ returns a maximal (for inclusion) element of

$$\left\{S | \forall k \in S.\, \text{protect}(k, K, \{t\})\right\}$$

There might be several maximal elements of this set. We, however, never encountered such a case in the protocols we studied. In order to decide satisfaction of the key protection constraint we would need to compute all these maximal elements. For simplicity reasons, we settled here for an underapproximation of this constraint choosing any maximal element.

In order to compute maxkeyprotect, we will need, as for the freshness constraint, to backtrack in the equational theory. The main problem here, as for freshness, is termination, as a rule like $f(c) \mapsto c$ might be quite annoying. In order to avoid this problem we will memoise the equations already encountered in a list l. As every constant const(_) has an antecedent without any constants in $\mathcal{C}_0$ as subterms the following algorithm is correct. We denote by $\mathcal{K}$ the set of all keys. We compute maxkeyprotect recursively using the following rules:

$$\text{maxkeyprotect}^l(\{m\}^r_k, K) =$$
$$\begin{cases} \mathcal{K} \text{ if } k \in K \\ \text{maxkeyprotect}^l(m, K) \cap \text{maxkeyprotect}(r, K) \text{ if } k \text{ is a key and } k \notin K \\ \text{maxkeyprotect}^l(m, K) \cap \text{maxkeyprotect}^l(k, K) \cap \text{maxkeyprotect}(r, K) \text{ otherwise} \end{cases}$$

In the CPA case the decryption is treated as any other function, we can then apply the following rule:

$$\text{maxkeyprotect}^l(\text{dec}(m, k), K) = \text{maxkeyprotect}^l(m, K) \cap \text{maxkeyprotect}^l(k, K)$$

In the CCA case, decrypting with a key is authorised, however we should be careful as anything protected by the decryption key under the decryption might be leaked. This leads to the following rule:

$$\text{maxkeyprotect}^l(\text{dec}(m, k), K) =$$
$$\begin{cases} \text{maxkeyprotect}^l(m, K \backslash k) \text{ if } k \text{ is a decryption key} \\ \text{maxkeyprotect}^l(m, K \cap \text{maxkeyprotect}^l(k, K)) \cap \text{maxkeyprotect}^l(k, K) \end{cases}$$

Note that in the previous case, $K \,\text{maxkeyprotect}^l(k, K)$ should be equal to K in all reasonable cases as K should be included in maxkeyprotect(k, K). The rule for the pair is the one we could expect:

$$\text{maxkeyprotect}^l(\text{pair}(u, v), K) = \text{maxkeyprotect}^l(u, K) \cap \text{maxkeyprotect}(v, K)$$

We consider the two cases where we have to backtrack according to the equational theory:

$$\text{maxkeyprotect}^l(\text{key}(i), K) =$$
$$\max(\{\text{maxkeyprotect}^{l::(t \mapsto \text{key}(i))}(t, K) | t \to \text{key}(i) \in E \backslash l\} \cup (\mathcal{K} \backslash \text{key}(i)))$$

$$\text{maxkeyprotect}^l(\text{sk}(i), K) =$$
$$\max(\{\text{maxkeyprotect}^{l::(t \mapsto \text{sk}(i))}(t, K) | t \mapsto \text{sk}(i) \in E \backslash l\} \cup (\mathcal{K} \backslash \text{sk}(i)))$$

If m is a constant in $\mathcal{C}_0$, we should find an antecedent that has no subterm in $\mathcal{C}_0$:

$$\text{maxkeyprotect}^l(\text{const}(i), K) =$$
$$\max\{\text{maxkeyprotect}^{l::(t \mapsto \text{const}(i))}(t, K) | t \mapsto \text{const}(i) \in E \backslash l\}$$

If we are computing the maximal protected key set of a handle, it is the maximal protected set of its dependencies, or, as a handle might be the right hand side of a rewriting rule, the maximal protected key set of its antecedents. Let us assume that the dependency set of handle(i) is S:

$$\text{maxkeyprotect}^l(\text{handle}(i), K) = \max\{\bigcap_{t\in S} \text{maxkeyprotect}(t, K), \\ \text{maxkeyprotect}^{l::(t\mapsto\text{handle}(i))}(t, K) | t \mapsto \text{handle}(i) \in E\backslash l\}$$

If c is constant in $\mathcal{C}$ that is not a (private) key or a handle, it protects all keys. This observation yields the following rule:

$$\text{maxkeyprotect}^l(c, K) = \mathcal{K} \text{ if } c \in \mathcal{C} \text{ and } c \text{ is not a key}$$

Note that, as for freshness we do not have to consider the rule of $E_{\mathcal{F}}$. Having shown how we can compute the maximal protected key set, it is enough to remark that a key is plaintext fresh if and only if it is protected by $\emptyset$. With this remark we can compute the pfresh constraint for free.

Key usability

In order to compute whether a key is usable or not, let us remark key usability is an inductive definition. A key is usable if it is protected by keys that are themselves usable. With these remarks, we compute the maximal set of usable keys of a set of terms S by computing the least fixed point of the following function:

$$F_S(K) = \bigcap_{t\in S} \text{maxkeyprotect}(t, K)$$

Computing this least fixed point terminates, as there might be only a finite number of keys that are not protected by a given set of terms.

5.5 Preprocessing and optimisations

In this section we present some simplifications of the original problem. First we have to get rid of the two axioms that may not be expressed as inference rules, namely the functionality and the reflexivity axiom. Indeed, these two axioms yield an infinite number of clauses, therefore we make sure that we include only the potentially useful instances, of which there are only finitely many.

5.5.1 Functionality and reflexivity

Besides computing a rewriting system for the equational theory, the preprocessing phase consists of adding the relevant instances of the functionality axiom. In the previous chapter, we do not consider neither functionality nor reflexivity axioms for the cases with constraints. This is because it is difficult to design a procedure when we have both constraints and such axioms. In this chapter, we choose another approach. We simply generate all relevant instances of functionality axioms: they involve only subterms of the equational theory or of the atoms. Let us state this formally:

Proposition 5.1. *Let R_E be a flat convergent rewriting system and S be a set of $\triangleright$ atoms (normalised with respect to R_E). Let us call $F(S, R_E)$ the set of instances $t_1, \ldots, t_n \triangleright t$ of* Fun *such that $t_1, \ldots, t_n, t$ are subterms of S or R_E. The following statements are equivalent:*

- *$S, R_E, E_{\mathcal{F}}$ is inconsistent with* Tr, Mon, Int, nm, Sec, Fresh, Refl *and* Fun
- *$S, F(S, R_E), R_E, E_{\mathcal{F}}$ is inconsistent with* Tr, Mon, Int, nm, Sec, Fresh *and* Refl.

Proof sketch. Let $t_1, \ldots, t_n \triangleright t$ be an instance of Fun that is not in F. Let us assume that $t_1, \ldots, t_n, t$ are in normal form for $R_E, E_{\mathcal{F}}$. If t is a subterm of S, R_E, then so are $t_1, \ldots, t_n$. Indeed, only the two following cases are possible:

- $t = f(t_1, \ldots, t_n)$ for some f and $f(t_1, \ldots, t_n)$ is irreducible. As t is a subterm of S, R_E, so are $t_1, \ldots, t_n$.
- There is a rule $f(t_1, \ldots, t_n) \to u$ in R_E, and we already have $t_1, \ldots, t_n$ are subterms of R_E.

Note that the rightmost branch of the proof is necessarily rooted in an atom in $S \cup F$, as the only way to derive a contradiction is using either the Fresh axiom of the nm axiom with an atom that has a constant as right hand side. It is now enough to remark that all rules have the following property: the right hand side of the left premise is a subterm of the left hand side of the right premise. Indeed the nm, Int or Tr rules have the following form:

$$\frac{t_1, \ldots, t_n \triangleright t \qquad C[t], u_1, \ldots, u_k \triangleright v}{A}\ \Psi$$

From this consideration and the fact that all Fun instances that are not in F have a right hand side that is not a subterm of S, R_E, we can conclude using a simple induction that all terms involved in the proof are subterms of S, R_E. □

Using proposition 5.1, we can compute all useful instances of the functionality axiom. A similar argument allows to consider only instances $t \triangleright t$ of Refl, such that $t \in \mathrm{st}(S, R_E)$. We call $R(S, R_E)$ the instances $t \triangleright t$ of reflexivity such that t is a subterm of S, R_E. Note that we do need the reflexivity axiom as, for example, an equational theory stating $A = n$ where A is a name and n is a nonce yields a contradiction. Indeed $n \triangleright n$ is a contradiction because A is a representative of n and $\mathrm{fresh}(n, A)$.

5.5.2 Optimisations

Unlike the decision procedure presented in chapter 4, the procedure implemented in this tool is not in **PTIME**. This is due to the fact that keeping track of constraints as presented in the previous chapter is not really practical. Moreover, we discovered during the experimentations with Scary that the key-usability constraint is necessary if we want to prove key-exchange protocols. However the soundness of the algorithm presented in chapter 4 relies heavily on the constraint being monotone. Considering that the algorithm presented here is not, in itself, particularly efficient, we need to make sure that we simplify the problem as much as possible.

Let us remark that adding axioms of the form $\emptyset \triangleright A$ for all names A and $\emptyset \triangleright \mathrm{pk}(i)$ for all public keys does not change the status of the problem. Indeed,

as public keys and names do not change the satisfaction of constraints, there is a proof of contradiction from S, E if and only if there is a proof of contradiction from $\overline{S}, E$ where $\overline{S}$ is the set of clauses obtained from S by removing all public keys and names from the left hand side of $\triangleright$ predicates. Adding the aforementioned axioms allows us to simplify the left hand side of $\triangleright$ atoms, removing the irrelevant information.

The second optimisation we present here relies on the two following observations:

- Using the functionality axiom and the Tr rule we easily see that

$$\mathrm{pair}(u, v); X \triangleright t \quad \leftrightarrow \quad u; v; X \triangleright t$$

- A constraint is satisfied by $\mathrm{pair}(u, v); S$ if and only if it is satisfied by $u; v; S$.

With these two observations, we know that replacing in pairs by their components in the left hand side of $\triangleright$ atoms is a sound simplification of the problem. This allows us to further reduce the number of terms involved in S.

5.6 The decision procedure

In the following section, we consider a set of $\triangleright$ atoms $\mathcal{S}$, an equational theory E (that we assume flat and convergent), and a set of disequations D that are the inputs to the decision procedure. We will also assume that all terms are in normal form with respect to the equational theory, and all constraints and matchings will be computed modulo E.

As in the previous chapter, we see the axioms as inference rules. Let us give the inference rules for the cryptographic axioms. We start by giving the rule for the two versions of the secrecy axiom:

$$\frac{X, \{x\}^r_{\mathrm{key}(i)} \triangleright \mathrm{nonce}(j)}{X \triangleright \mathrm{nonce}(j)} \, \mathrm{usable}^{\mathrm{CCA}}(\mathrm{key}(i), (X, x)), \mathrm{fresh}(r, (X, x))$$

$$\frac{X, \{x\}^r_{\mathrm{pk}(i)} \triangleright \mathrm{nonce}(j)}{X \triangleright \mathrm{nonce}(j)} \, \mathrm{usable}^{\mathrm{CCA}}(\mathrm{sk}(i), (X, x)), \mathrm{fresh}(r, (X, x))$$

Given a $\triangleright$ atom $S, u \triangleright t$ and a key k, deciding whether this rule is applicable amounts only to

- check whether t is a nonce or not: it is enough to choose and ordering in which nonces are minimal to ensure that t is a nonce if and only if its normal form is a nonce.
- check whether u can be matched by $\{x\}^y_k$ with the constraint $\mathrm{usable}(k, (S; x))$

Let us now give the rules for integrity and non-malleability:

$$\frac{X \triangleright y \quad X', \mathrm{adec}(y, sk) \triangleright n}{\bigvee_{\{x\}^r_{pk} \in st(X)} y = \{x\}^r_{pk}} \, C$$

with C the conjunction of the following constraints:

- $\mathrm{usable}^{\mathrm{CCA}}(sk, X)$
- n only appears under encryption by X usable keys

Again, given two premises of this rule it is easy to check whether they match the rule and compute the conclusion. The rule for non malleability is very similar:

$$\frac{X \rhd y \quad Y, C[y] \rhd z}{Y \rhd z \quad \vee \bigvee_{\{z\}^r_{\mathrm{key}(i)} \in \mathrm{st}(X)} y = \{z\}^r_{\mathrm{key}(i)}} \mathrm{usable}^{\mathrm{CCA}}(\mathrm{key}(i), X)$$

Now, let us go into the decision procedure in itself. First of all, we need to get rid of the functionality and transitivity rules. This is done thanks to the preprocessing phase. We start by rephrasing the problem of deciding whether $\mathcal{S}$, E, D is consistent with $\mathcal{A}$. Let us assume that $\mathcal{S}$ contains $F(S, R_E)$ and $R(S, R_E)$. Thanks to proposition 5.1, the problem is equivalent to the existence of $\mathcal{S}'$, E', D' such that

- $\mathcal{S} \subseteq \mathcal{S}'$, $E \subseteq E'$, $D \subseteq D'$
- $\mathcal{S}'$, E', D' is invariant under application of the rules for transitivity, secrecy, non-malleability and integrity.
- $\mathcal{S}'$ does not contain any $S \rhd t$ that satisfies the premises from the freshness rule
- for all $u \neq t$ in D', we have $\neg(u =_{E'} t)$

In order to check whether such a $\mathcal{S}', E', D'$ exists, we complete a two steps saturation procedure. We start by saturating with the secrecy and transitivity axiom,

- for $S \rhd t$ atom in $\mathcal{S}$, we add to $\mathcal{S}$ all possible consequences of the secrecy axiom with premise $S \rhd t$
- for every $S_1 \rhd t_1$ and $S_2 \rhd t_2$ in $\mathcal{S}$ we add $S_1; (S_2 \backslash t_1) \rhd t_2$ to $\mathcal{S}$ (i.e. we apply the transitivity rule)

For this first part, the equations and disequations are unchanged. Assuming the problem is saturated with the first part of the algorithm, we saturate with the non-malleability and unforgeability rules:

- let $S_1 \rhd t_1$ and $S_2 \rhd t_2$ be in $\mathcal{S}$. If the non-malleability rule applies, let L be the list of equations that conclude the rule. If there exists an equation e in L such that e is a consequence of E, proceed to the following pair of atoms in $\mathcal{S}$. Otherwise for each equation $e_1 = e_2$ in L:
  - if $E \cup (e_1 = e_2)$ is compatible with $\mathcal{D}$, try to run the saturation with $\mathcal{S}, E \cup (e_1 = e_2), D$. If it returns a model, then return this model and escape the main procedure. Otherwise add $e_1 \neq e_2$ to D
  - otherwise add $e_1 \neq e_2$ to D

  if no model has been found, then $\mathcal{S}, E, D$ is inconsistent with the axioms, return unsatisfiable.
- let $S_1 \rhd t_1$ and $S_2 \rhd t_2$ be in $\mathcal{S}$. If the non-malleability rule applies, let L be the list of equations that conclude the rule and $S \rhd t$ be the $\rhd$ atom that concludes the rule. If there exists an equation e in L such that e is a consequence of E, proceed to the following pair of atoms. Otherwise for each equation $e_1 = e_2$ in L:
  - if $E \cup (e_1 = e_2)$ is compatible with $\mathcal{D}$ try to run the saturation with $\mathcal{S}, E \cup (e_1 = e_2), D$. If it return a model, then return this model and escape the main procedure. Otherwise add $e_1 \neq e_2$ to D
  - otherwise add $e_1 \neq e_2$ to D

  if no model has been found, then add $S \rhd t$ to $\mathcal{S}$

Note that this saturation procedure terminates since there is a bound on the number of equations and a bound on the number of clauses that can be added. Indeed all terms are subterms of the original $\mathcal{S}$ or E.

5.7 Experimental results

The first benchmark is the NSL protocol. We found automatically the attack presented in [BAS12]. This attack on one honest session plus one session with the attacker relies on an unusual identity. It works when the second projection of a nonce n may be equal to the name of the intruder I. If we add the disequation $\pi_2(n) \neq I$, then NSL (when limited to two sessions) is proved secure with our tool. A simplification of this trace is the one presented in section 5.2.

$$\begin{array}{lcl} A & \rightarrow & \{n_b\}_{pk_b} \\ B & \leftarrow & \{x_n, I\}_{pk_b} \\ B & \rightarrow & \{x_n, n', B\}_{pk_I} \end{array}$$

A maybe more interesting point is the way we recover the actual attack from the model. Let us give an excerpt of the output of SCARY on this trace:

```
constant(0) = nonce(2)
pi2(nonce(2)) = name(3)
adec(handle(1),sk(2)) = nonce(2)
aenc(nonce(2),pk(2),nonce(123)) = handle(1)
```

From this, we clearly see that in the model found there is the equations $\pi_2(n_b) = I$. We also see that the received message represented by handle(1) is indeed a replay of the first message. With this observation, we can check that this is an actual attack on the protocol. Indeed, if the equation $\pi_2(n_B) = I$, forwarding the first message to B yields an attack.

We also tested our tool on various protocols (Wide Mouth Frog, BAN-Yahalom, Andrew Secure RPC, NSL, Woo and Lam Pi, ...) involving only asymmetric encryption, which were proven secure (not finding new attacks).

The second experimental result, is that we discovered that the plaintext freshness that was used in the original secrecy and integrity axioms were not permissive enough to prove most symmetric protocols, as keys are sent around. Typically the problem was that we want to use the integrity rule on $S \triangleright t$ and $S', \mathrm{dec}(t,k) \triangleright n$ but, as t is not a nonce, we can not remove the encrypted instances of k from S. We also modified the original integrity rule that was

$$\frac{X \triangleright x \qquad Y, \mathrm{dec}(x,k) \triangleright n}{X, Y \triangleright n \vee \bigvee_{\{z\}_k^r \in \mathrm{st}(X)} y = \{z\}_k^r} C$$

by adding a context on top of $\mathrm{dec}(x,k)$ as we want to apply integrity even for terms that may not be at top level and removing the Y in the conclusion.

The third experimental result is an attack we discovered on Andrew's secure RPC, when the shared key is used to encrypt a secret. Recall the protocol:

$$\begin{array}{lcl} A & \rightarrow & B : A, \{N_a\}_{Kab} \\ B & \rightarrow & A : \{succ(N_a), N_b\}_{Kab} \\ A & \rightarrow & B : \{succ(N_b)\}_{Kab} \\ B & \rightarrow & A : \{K'_{ab}, N'_b\}_{Kab} \\ A & \rightarrow & B : \{s\}_{K'ab} \end{array}$$

As we do not have a successor function, we encoded $succ(x)$ as $\text{pair}(x, x)$. The attack relies on the fact that we may force an honest agent to encrypt a message with the second projection of a nonce (for example). As the last message containing the key is not related in any way with the rest of the session, the adversary can replay the first message sent out by A, thus obtaining the encryption of the secret with the first projection of a nonce, which might not be secure. Such an attack is not found by classical symbolic protocol analysis.

5.8 Further work

SCARY is still a work in progress. As the example of Andrew's secure RPC shows, it would be interesting to add free functions to our input syntax, in order to model primitives like the successor function. Moreover, it would also be interesting to add support for equational theories for these functions. Remark that the algorithm we presented in section 5.4 could be extended to a larger class of equational theories.

Adding new primitives and new axioms may yield significant modifications of the strategies, in order to keep a reasonable efficiency. Though, as soon as the axioms are proven sound, there is no obstacle to such extensions. For instance, adding signatures should be easy.

Another important extension would be adding support for more properties than simple secrecy. For example it is not difficult to add agreement properties. This would not modify the core decision procedure on clauses. It would only involve extending the module of the tool that generates clauses from a protocol.

Chapter 6

Conclusion

In this thesis we showed that the Computationally Complete Symbolic Attacker framework can be automated. Our first contribution is giving a simple formulation of the computational semantics of first order logic, based on ideas from Bana and Okada. This simpler formulation yields simpler soundness proofs for the axioms. These proofs are also easier to check. We also provided, in chapter 3, axioms that are strong enough to prove secure most protocols involving symmetric and asymmetric encryption. These axioms are more general than the ones presented in [BC12]. This generalisation is crucial for proving protocols involving symmetric encryption in a non-trivial way. We also fixed a gap in the proofs of the secrecy axiom. We therefore demonstrated that this model can be made powerful enough to actually prove protocols secure. We should also note that, compared to usual computational soundness proofs, the soundness proofs of the axioms are much simpler.

In a second part in Chapters 4 and 5, we showed that it is theoretically and practically feasible to automate this model. The **PTIME** decision procedure proposed in chapter 4 shows that the first order satisfiability problems involved in proving security with respect to the Computationally Complete Symbolic Attacker are far from untractable if the axioms reflect either a secrecy property or an integrity property. Our tool Scary provides with an implementation of (a variant of) this decision procedure. While we loose the **PTIME** complexity in this tool, we showed that it is still efficient enough to prove actual cryptographic protocols. In addition we also found a new attack on the protocol Andrew's Secure RPC, and hope to find more attacks using this tool in the future.

Having demonstrated that this model is powerful enough to prove protocols with very accurate assumptions and that it also is automatable leads us to various lines of research.

6.1 Extend the model and tool

In this thesis we only considered two cryptographic primitives: symmetric and asymmetric encryption. Having shown that this model leads to interesting results with only these two primitives leads us to think that it would be feasible and beneficial to extend it to other primitives. Extending the model to other primitives would involve both finding relevant computationally sound axioms

for these primitives and extending the decision procedure and the tool to cope with such axioms.

The most widely used cryptographic primitive, besides encryption is signature. As signature ensures integrity, devising sound axioms for it and adding it to Scary would be reasonably simple. Blind signature and homomorphic encryption are two primitives that are usually hard to deal with in the symbolic model. This is due to the fact that their equational theories are much more complicated than the equational theory of encryption. However, the cryptographic properties of these primitives are secrecy and integrity properties. Considering that in the Computationally Complete Symbolic Attacker framework we may add only the part of the equational theory that is relevant to the protocol proof, we would not get the difficulties of other tools. Therefore it is also reasonable to think that these primitives might be added to our model at a relatively low cost, both in terms of proofs and of complexity of the decision procedure. Moreover, this approach of adding only the necessary properties could let us cope with XOR and overcome Backes and Pfitzmann's impossibility result[BP05].

For other primitives like hash functions or zero-knowledge proofs, the task is more complicated. However the model is permissive enough to allow us hope in modelling these primitives. The hard part would be to extend the decision procedure to cope with such axioms. There is nonetheless hope that our approach consisting of transforming axioms into inference rules would allow us to obtain a decision procedure for such axioms.

6.2 Equivalence properties

A similar model has been introduced very recently in [BCL14], for equivalence properties. The predicate symbol $\triangleright$ is replaced there with an indistinguishability predicate. It remains however, for this logic, to design a library of sound axioms (as we did in chapter 3) and to design and implement a verification tool.

6.3 Compositionality

Another open question is the question of composition of protocols. If a protocol P involves a key-exchange, it is reasonable to see the key exchange protocol as a subroutine of the protocol. We would like to perform a proof of P that does not rely on the particular key exchange used in the protocol but on the fundamental properties that a key exchange should satisfy.

Note that the computational semantics of the Computationally Complete Symbolic Attacker allows us to model the attacker's power as we wish. Therefore, we could consider doing a proof of P (without the key exchange) against an attacker that has access to the key exchange protocol satisfying some property Ψ as an oracle. The proof of P would then hold for any key exchange satisfying this property Ψ. In other terms we would have to prove that:

- the key exchange K satisfies Ψ (with a PPT attacker)
- the protocol P satisfies the security property with an attacker accessing an oracle for the key exchange

in order to get the security of $P(K)$ against a PPT attacker. This idea is quite similar to the key usability notion proposed in [DDMW06].

6.4 Other attacker models

One consequence of the freedom that the Computationally Complete Symbolic attacker leaves for specifying the attacker's power is that it would not be too hard to consider more powerful attackers. More precisely one could consider attacker having access to side channels, for example timing channels or power consumption channels. Modelling such side channels would involve some changes in the model as these side channels depend on the run of the protocol and not only on the messages that are sent out. However, extending the observations of the attacker to the tests performed during a run of the protocol leaves some hope of extending the model to an attacker that has access to side channels.

In a broader perspective, we believe that ideas from the Computationally complete symbolic attacker could be used for proving security in much more diverse context. For example security against fault injection attacks. Indeed the approach of specifying the minimal properties that an attacker may not break, and trying to prove that these are inconsistent with the negation of the security property is quite general. Moreover, the computational semantics presented in chapter 2 could easily be extended for various attacker models.

Bibliography

[ABB+05] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, pages 281–285. Springer, 2005.

[ACC+08] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuéllar, and M. Llanos Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In Vitaly Shmatikov, editor, *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering, FMSE 2008, Alexandria, VA, USA, October 27, 2008*, pages 1–10. ACM, 2008.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In Chris Hankin and Dave Schmidt, editors, *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 104–115. ACM, 2001.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptology*, 15(2):103–127, 2002.

[BAS12] Gergei Bana, Pedro Adão, and Hideki Sakurada. Computationally complete symbolic attacker in action. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 546–560. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[BC12] Gergei Bana and Hubert Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In Pierpaolo Degano and Joshua D. Guttman, editors, *Principles of Secu-*

rity and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings, pages 189–208. Springer, 2012.

[BCL14] Gergei Bana and Hubert Comon-Lundh. A computationally complete symbolic attacker for equivalence properties. In *Proc. ACM Conference on Computers and Communications Security*, 2014.

[BCW13] Florian Böhl, Véronique Cortier, and Bogdan Warinschi. Deduction soundness: prove one, get five for free. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1261–1272. ACM, 2013.

[BFCZ12] Karthikeyan Bhargavan, Cédric Fournet, Ricardo Corin, and Eugen Zalinescu. Verified cryptographic implementations for TLS. *ACM Trans. Inf. Syst. Secur.*, 15(1):3, 2012.

[BG01] David A. Basin and Harald Ganzinger. Automated complexity analysis based on ordered resolution. *J. ACM*, 48(1):70–109, 2001.

[BGHB11] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 71–90. Springer, 2011.

[BGLB11] Gilles Barthe, Benjamin Grégoire, Yassine Lakhnech, and Santiago Zanella Béguelin. Beyond provable security verifiable IND-CCA security of OAEP. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 180–196. Springer, 2011.

[BHO13] Gergei Bana, Koji Hasebe, and Mitsuhiro Okada. Computationally complete symbolic attacker and key exchange. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1231–1246. ACM, 2013.

[BKY01] Enrico Buonanno, Jonathan Katz, and Moti Yung. Incremental unforgeable encryption. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2001.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*, pages 82–96. IEEE Computer Society, 2001.

[Bla06] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *2006 IEEE Symposium on Security and Pri-*

vacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA, pages 140–154. IEEE Computer Society, 2006.

[BMU12] Michael Backes, Ankit Malik, and Dominique Unruh. Computational soundness without protocol restrictions. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 699–711. ACM, 2012.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4):469–491, 2008.

[BP04] Michael Backes and Birgit Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004), 28-30 June 2004, Pacific Grove, CA, USA*, pages 204–218. IEEE Computer Society, 2004.

[BP05] M. Backes and B. Pfitzmann. Limits of the cryptographic realization of Dolev-Yao style XOR. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS)*, 2005.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 92–111. Springer, 1994.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2002.

[CC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 109–118. ACM, 2008.

[CCD10] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Automating security analysis: Symbolic equivalence of constraint systems. In Jürgen Giesl and Reiner Hähnle, editors, *Automated Reasoning, 5th International Joint Conference, IJCAR 2010, Edinburgh, UK, July 16-19, 2010. Proceedings*, pages 412–426. Springer, 2010.

[CCS12] Hubert Comon-Lundh, Véronique Cortier, and Guillaume Scerri. Security proof with dishonest keys. In Pierpaolo Degano and Joshua D. Guttman, editors, *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software,*

ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings, pages 149–168. Springer, 2012.

[CHKS12] Hubert Comon-Lundh, Masami Hagiya, Yusuke Kawamoto, and Hideki Sakurada. Computational soundness of indistinguishability properties without computable parsing. In Mark Dermot Ryan, Ben Smyth, and Guilin Wang, editors, *Information Security Practice and Experience - 8th International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings*, pages 63–79. Springer, 2012.

[Cre08] Cas J. F. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In Aarti Gupta and Sharad Malik, editors, *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, pages 414–418. Springer, 2008.

[CT97] Hubert Comon and Ralf Treinen. The first-order theory of lexicographic path orderings is undecidable. *Theor. Comput. Sci.*, 176(1-2):67–87, 1997.

[DDM+05] Anupam Datta, Ante Derek, John C. Mitchell, Vitaly Shmatikov, and Mathieu Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 16–29. Springer, 2005.

[DDMW06] Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy*, pages 321–334. IEEE Computer Society, 2006.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[Fit70] Melvin Fitting. An embedding of classical logic in S4. *J. Symb. Log.*, 35(4):529–534, 1970.

[FKS11] Cédric Fournet, Markulf Kohlweiss, and Pierre-Yves Strub. Modular code-based cryptographic verification. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 341–350. ACM, 2011.

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *J. Cryptology*, 17(2):81–104, 2004.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[HMA+08] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Collision-based power analysis of modular exponentiation using chosen-message pairs. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 15–29. Springer, 2008.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113. Springer, 1996.

[KR05] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In Shmuel Sagiv, editor, *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, pages 186–200. Springer, 2005.

[Low95] Gavin Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.

[McA93] David A. McAllester. Automatic recognition of tractability in inference relations. *J. ACM*, 40(2):284–303, 1993.

[MW04] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway language of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004.

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[Poi05] David Pointcheval. *Advanced Course on Contemporary Cryptology*, chapter "Provable Security for Public-Key Schemes", pages 133–189. Advanced Courses CRM Barcelona. Birkhäuser Publishers, Basel, 2005.

[RDM07] Arnab Roy, Anupam Datta, and John C. Mitchell. Formal proofs of cryptographic security of diffie-hellman-based protocols. In Gilles Barthe and Cédric Fournet, editors, *Trustworthy Global Computing, Third Symposium, TGC 2007, Sophia-Antipolis, France, November 5-6, 2007, Revised Selected Papers*, pages 312–329. Springer, 2007.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara,*

California, USA, August 11-15, 1991, Proceedings, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[Sat89] Mahadev Satyanarayanan. Integrating security in a large distributed system. *ACM Trans. Comput. Syst.*, 7(3):247–280, 1989.

[Sho02] Victor Shoup. OAEP reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[Sin99] Simon Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*. Doubleday, New York, NY, USA, 1st edition, 1999.

[SPO14] Spore — security protocols open repository. `http://www.lsv.ens-cachan.fr/Software/spore/index.html`, 2014. [Online; accessed 8-October-2014].

[THG99] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1):191–230, 1999.

[War03] Bogdan Warinschi. A computational analysis of the Needham-Schröeder-(Lowe) protocol. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003), 30 June - 2 July 2003, Pacific Grove, CA, USA*, page 248. IEEE Computer Society, 2003.

[wik14] Cryptanalysis of the enigma — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Cryptanalysis_of_the_Enigma`, 2014. [Online; accessed 8-October-2014].