Sujet de stage M2
Méthodes symboliques pour la bi-déduction dans le contexte de la vérification de protocoles de sécurité

Laboratoire & institution : IRISA, CNRS & Univ Rennes, Rennes, France

Équipe : SPICY – https://spicy.irisa.fr

Encadrants :

— David Baelde, david.baelde@irisa.fr

— Stéphanie Delaune, stephanie.delaune@irisa.fr

Contexte général

Alors qu'une part croissante des activités humaines se fait via des communications électroniques (messagerie, navigation web, paiements, vote, ...), les protocoles visant à sécuriser ces échanges prennent une importance grandissante. La problématique générale de la vérification formelle des systèmes informatiques se décline de façon un peu particulière quand on considère les protocoles de sécurité, et a été le sujet de recherches actives depuis les années 1970 [CK14]. Aujourd'hui, plusieurs types d'outils permettent d'analyser des protocoles avec l'aide de l'ordinateur ; ils diffèrent notamment par le niveau d'abstraction adopté, le degré d'automatisation, l'aptitude à trouver des attaques et/ou fournir des preuves.

L'outil Squirrel [Squ21] est un assistant de preuve dédié à la preuve de protocoles, et apportant un équilibre intéressant entre automatisation et précision par rapport aux autres outils existants. L'approche théorique sous-jacente est fondée sur une idée proposée par G. Bana et H. Comon en 2012. Cette idée a évolué et a été validée sur plusieurs études de cas concrètes, mais ce n'est que relativement récemment que le cadre théorique a évolué et enfin pu être implémenté pour permettre des preuves vérifiées par ordinateur pour des exécutions non bornées de protocoles [BDJ+21]. Datant de 2021, l'outil Squirrel est bien entendu encore sujet à de nombreuses évolutions dont certaines sont d'ailleurs en développement au sein de l'équipe Spicy.

La logique de l'outil Squirrel est construite pour manipuler facilement des propriétés d'équivalence, c'est-à-dire d'indistinguabilité entre plusieurs systèmes. Un raisonnement particulièrement utile pour mener à bien des preuves dans ce domaine est la notion bi-déduction utilisée en particulier pour faire des preuves par réduction. Ce problème de la bi-déduction, noté $(u, v) \triangleright_{\mathsf{E}} (u', v')$ et motivé par la preuve de protocoles, est utile en particulier dans l'outil Squirrel. Il peut se fomuler de la façon suivante :

Entrées : Deux paires de termes (u, v) et (u', v'), et une théorie équationnelle E, construites sur une signature $\mathcal{F}$.

Sortie : Est-ce qu'il existe un contexte $C[\_]$, i.e. un terme construit sur $\mathcal{F}$ avec un « trou », tel que $C[u] =_{\mathsf{E}} u'$ et $C[v] =_{\mathsf{E}} v'$?

Les termes u, v, u', et v' peuvent être supposés clos, i.e. sans variable. Ce problème peut être vu comme un cas particulier d'unification du second ordre modulo une théorie équationnelle E. Le problème d'unification du second-ordre est connu pour être indécidable en général, mais le

problème nous intéressant ici, est un cas particulier et est connu pour être décidable e.g. si E est la théorie vide.

En réalité, SQUIRREL se situe dans le modèle standard des cryptographes, où l'on raisonne sur des attaquants modélisés par des machines de Turing (probabilistes et en temps polynomial). La bi-déduction utilisée dans SQUIRREL est donc nettement plus complexe que la définition précédente : grossièrement, il faut remplacer le contexte C par une machine de Turing arbitraire ! L'objectif n'est pas de remettre en question la généralité de l'approche sous-jacente à SQUIRREL, mais d'étudier une version simplifiée du problème dans un cadre facilitateur, dans l'espoir de développer des algorithmes qui puissent ensuite être transférés dans le cadre général ; l'espoir est d'obtenir ainsi de meilleures techniques que celles actuellement développées de façon ad-hoc. Plus généralement, les techniques de preuve utilisées en SQUIRREL sont bien sûr toujours prouvées correctes, mais les résultats de complétude dans ce cadre sont quasi inexistants — la seule exception publiée, à notre connaissance, est un petit résultat portant sur les égalités entre termes composés uniquement de constructions $\mathsf{if}\ \cdot\ \mathsf{then}\ \cdot\ \mathsf{else}\ \cdot$, constantes booléennes et variables [BCEO19].

Objectif du stage

Un premier objectif pour le stage consistera à se familiariser avec la litérature sur le sujet, en particulier avec les notions de déduction et d'indistinguabilité (appelée aussi équivalence statique). Ces deux notions ont été étudiées par le passé pour différentes théories équationnelles particulièrement pertinentes dans le contexte de la vérification de protocoles [AC06], e.g. les théories sous-termes convergentes permettant de modéliser les mécanismes de chiffrements et de signatures, la théorie permettant de représenter l'opérateur de ou-exclusif, ...

L'objectif principal du stage consistera en l'étude du problème de la bi-déduction. Celui-ci pourra être étudié pour différentes (classes de) théories équationnelles, en gardant en ligne de mire l'application à la bi-déduction dans SQUIRREL. En plus des théories modélisant les primitives cryptographiques, on cherchera à traiter des théories portant sur les opérateurs natifs de l'outil, e.g. $\mathsf{if}\ \cdot\ \mathsf{then}\ \cdot\ \mathsf{else}\ \cdot$, $\mathsf{diff}(\cdot,\cdot)$, $\cdot = \cdot$, ...

Ensuite, en fonction du temps restant et de vos envies, d'autres questions pourront être abordées pendant le stage :

- *Modularité* : est-il possible de mettre au point une méthode de combinaison permettant à partir de deux algorithmes de bi-déduction $\mathcal{A}_1$ and $\mathcal{A}_2$ pour des théories E_1 and E_2 d'obtenir un algorithme pour la bi-déduction pour la théorie $\mathsf{E}_1 \cup \mathsf{E}_2$? Une telle méthode existe, e.g. pour le problème de la déduction lorsque les théories sont disjointes [CD12], et pourrait peut-être être étendue à la bi-déduction.
- *Implémentation dans l'outil* SQUIRREL[1] : les algorithmes mis au point pourront être implémentés et ainsi remplacer la procédure ad-hoc et incomplète utilisée à l'heure actuelle. Des études de cas pourront ensuite être réalisées pour évaluer le gain en pratique.
- *Complètude par rapport au modèle calculatoire* : une fois le problème résolu dans le modèle symbolique, on pourra se demander sous quelles conditions (et en quel sens) la solution obtenue est complète vis à vis de la formulation de la bi-déduction dans le modèle calculatoire (i.e. avec des machines de Turing) ; des résultats négatifs ne sont pas à exclure, et seraient néanmoins très intéressants.

Ce stage pourra par la suite conduire à une thèse sur des sujets similaires et pourra être financé sur des projets en cours dans l'équipe.

1. Outil SQUIRREL : https ://squirrel-prover.github.io

Compétences attendues : Des bases solides en informatique fondamentale (e.g. logique, réécriture) sont attendues. En revanche, des connaissances en sécurité ne sont pas nécessaires et pourront être acquises pendant le stage.

Références

[AC06] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2) :2–32, November 2006.

[BCEO19] Gergei Bana, Rohit Chadha, Ajay Kumar Eeralla, and Mitsuhiro Okada. Verification methods for the computationally complete symbolic attacker based on indistinguishability. *ACM Transactions on Computational Logic (TOCL)*, 21(1) :1–44, 2019.

[BDJ+21] D. Baelde, S. Delaune, C. Jacomme, A. Koutsos, and S. Moreau. An interactive prover for protocol verification in the computational model. In Alina Oprea and Thorsten Holz, editors, *Proceedings of the 42nd IEEE Symposium on Security and Privacy (S&P'21)*, San Francisco, California, USA, May 2021. IEEE Computer Society Press.

[CD12] V. Cortier and S. Delaune. Decidability and combination results for two notions of knowledge in security protocols. *Journal of Automated Reasoning*, 48(4) :441–487, April 2012.

[CK14] V. Cortier and S. Kremer. Formal models and techniques for analyzing security protocols : A tutorial. *Found. Trends Program. Lang.*, 1(3) :151–267, 2014.

[Squ21] Squirrel. https://squirrel-prover.github.io, 2021.