Communication entre Easycrypt et les prouveurs automatiques
Thématique : Typage/cryptologie

Laboratoire : INRIA Sophia-Antipolis
Ville : Nice
Équipe : Marelle
Directeur de stage : Benjamin Gregoire, `Benjamin.Gregoire@inria.fr`
Directeur du laboratoire : Gerard Giraudon, `Gerard.Giraudon@sophia.inria.fr`

Présentation générale du domaine

L'avènement des prouveurs SMT puissants et des générateurs de condition de vérification efficaces a permis de faire de la vérification déductive de programmes une technologie efficace pour prouver la sécurité et correction de programmes complexes. Cependant la vérification de propriétés de sécurité reste un challenge significatif pour deux raisons. Premièrement, les propriétés de sécurités sont généralement exprimées comme une propriété entre deux programmes. Deuxièmement une classe importante de programmes, comme les protocoles cryptographiques, effectue des calculs probabilistes qui sont en dehors du domaine d'application des outils de vérification actuels. development team Easycrypt[1] est un outil permettant la vérification de primitives et de protocoles cryptographiques. Easycrypt capture les méthodes de raisonnement couramment utilisées dans les preuves cryptographiques grâce à la logique de Hoare relationnelle probabiliste[2] (pRHL) qui est une extension au cas probabiliste de la logique de Hoare relationnelle[3]. Les jugements sont de la forme :

$$\vdash c_1 \sim c_2 : \psi \Rightarrow \phi$$

avec c_1, c_2 des programmes probabilistes, ψ une précondition (une relation) sur les état initiaux et ϕ une postcondition sur les états finaux. La validité des jugements de cette logique permet de justifier des propriétés de sécurité, exprimées en terme de probabilité, de la forme

$$\Pr[c_1, m_1 : E_1] <= \Pr[c_2, m_2 : E_2]$$

où m_1 et m_2 sont des états initiaux satisfaisant la précondition ψ et E_1 et E_2 sont des événements relié par la postcondition ϕ.

Objectifs du stage

Comme toute logique de Hoare, pRHL contient des jugements purement logiques $\vdash \phi$ où ϕ est une formule du premier ordre. Easycrypt utilise les prouveurs

SMT pour établir la validité de ces jugements. La communication avec les prouveurs ce fait à l'aide de l'outil Why3[5]. Malheureusement ces prouveurs ne sont pas infaillibles (en particulier sur les formules avec quantificateurs). L'objectif général de ce stage est d'améliorer l'interaction avec les prouveurs automatiques dans Easycrypt.

La première étape consistera à développer un mini-prouveur interactif au dessus de Why3. Pour cela il faudra implémenter les règles de base de la logique telles que l'introduction ou l'élimination des connecteurs logiques. L'idée étant de permettre de faire des étapes de preuve élémentaires tout en permettant à tout moment l'utilisation des prouveurs automatiques. Ceci sera particulièrement utile pour les formules avec quantificateurs. Dans un deuxième temps on pourra rajouter des règles de logique plus compliquées telles que les schémas d'inductions qui peuvent être automatiquement engendrés sur les types concrets.

La seconde étape consistera à définir une théorie logique (au sens de Why3) pour les groupes cycliques. Il s'agira de comprendre comment axiomatiser les opérations élémentaires afin de permettre aux prouveurs automatiques de résoudre le maximum de lemmes sur les groupes cycliques.

Si le temps le permet la dernière étape consistera à améliorer la communication entre Why3 et les prouveurs automatiques. En effet, la logique de Why3 (polymorphisme, données et prédicats inductifs) est plus riche que la logique implémentée par la plupart des prouveurs. Why3 implémente différentes techniques pour traduire sa logique vers les logiques des prouveurs. Il s'agira de comprendre si les travaux de J. Blanchette [4] permettraient d'améliorer la communication de Why3 avec les différents prouveurs.

Compétences espérées

Programation OCaml, quelques notions de cryptographies et de logique, compréhension des systèmes de type polymorphes.

Références

[1] Barthe, G., Grégoire, B., Heraud, S., Béguelin, S.Z. : Computer-aided security proofs for the working cryptographer. In : Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 71–90. Springer (2011), best Paper Award

[2] Barthe, G., Grégoire, B., Zanella, S. : Formal certification of code-based cryptographic proofs. In : 36th symposium Principles of Programming Languages.

ACM Press (Jan 2009), http://www-sop.inria.fr/everest/personnel/Benjamin.Gregoire/Publi/popl09.pdf

[3] Benton, N. : Simple relational correctness proofs for static analyses and program transformations. In : 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 14–25 (Jan 2004)

[4] Blanchette, J.C., Böhme, S., Smallbone, N. : Monotonicity or how to encode polymorphic types safely and efficiently, submitted

[5] The Why3 development team : The why3 platform. Online –http://why3.lri.fr/