Preuves de protocoles cryptographiques : méthodes symboliques et attaquants puissants

Charlie Jacomme [1]

Charlie Jacomme a soutenu sa thèse [2] le 16 octobre 2020 à l'université Paris Saclay, sous la direction de Hubert Comon et Steve Kremer. Vous trouverez ci-dessous un résumé de cette thèse.

Nos ordinateurs, téléphones et autres objets connectés font maintenant partie intégrante de nos vies. Nous les utilisons pour de nombreuses tâches, que ce soit pour communiquer avec nos proches, faire des achats en ligne, prendre des rendez-vous médicaux... Ces systèmes informatiques manipulent alors un grand nombre de nos données personnelles.

Si certaines fuites de données vont clairement à l'encontre de nos intérêts, par exemple lorsqu'un tiers peut obtenir nos informations de carte bleue, les fuites de données telles que notre position ou nos messages représentent des atteintes moins évidentes à la vie privée. On trouve parmi les exemples les plus dramatiques le récent suivi des Ouïghours via leurs téléphones par les autorités chinoises. Il existe aussi des exemples plus subtils : par exemple où une personne peut se voir refuser un prêt car la compagnie d'assurance a pu découvrir qu'elle avait une longue maladie. Il existe ainsi de nombreux exemples où nos données personnelles sont utilisées à

1. charlie.jacomme@cispa.de.
2. https://theses.fr/2020UPASG005.

DOI:10.48556/SIF.1024.19.93

l'encontre de nos intérêts, que ce soit par une personne tierce, un gouvernement [3] ou une entreprise. Une question essentielle se pose alors :

Pouvons-nous avoir des garanties sur l'accès et l'utilisation de nos données ?

La réponse à cette question est aujourd'hui non, et trois aspects doivent être étudiés pour changer cette réponse :

(1) mettre au point des systèmes visant à protéger la vie privée de leurs utilisateurs ;

(2) s'assurer que ces protections sont efficaces ;

(3) et faire en sorte que ces systèmes soient déployés à grande échelle.

Malheureusement, si les entreprises s'intéressent aux questions de sécurité, elles s'intéressent souvent peu aux questions de vie privée, et ce seulement lorsque cela peut compromettre leurs intérêts économiques. Ainsi, il appartient à la recherche publique de se saisir de ces questions, en développant, vérifiant et rendant accessibles des systèmes respectueux de la vie privée, et c'est ce qui m'a intéressé lors de ma thèse. Je me suis intéressé plus particulièrement à l'une des grandes questions scientifiques autour de ce sujet :

Comment fournir des garanties de respect de la vie privée par les systèmes informatiques ?

Contexte scientifique

Les systèmes de communication reposent sur des protocoles de sécurité, i.e. des programmes distribués qui visent à permettre la communication de manière sécurisée entre plusieurs agents. Parmi les protocoles les plus connus et utilisés, on trouve TLS (utilisé pour les connections *https*), SSH ou encore AKA pour les communications téléphoniques.

Ces protocoles reposent souvent sur des primitives cryptographiques telles que le chiffrement (RSA, AES...) ou les fonctions à sens unique (MD5, SHA-3...), qui sont des constructions mathématiques permettant notamment de masquer le contenu d'un message. Par exemple, étant donné une clé secrète sk_A et sa clé publique associée pk_A, une fonction de chiffrement asymétrique `enc` permet à toute personne possédant la clé publique de chiffrer des messages de telle sorte que seule la personne possédant la clé secrète puisse les déchiffrer. Équipé d'une telle primitive, on peut alors imaginer un protocole d'échange de clé entre un initiateur I et un répondeur R qui souhaitent se mettre d'accord sur un secret commun. Pour ce faire, ils peuvent procéder comme illustré dans la figure 1, où chacun tire au hasard une suite de bits dénotée e_X et chiffre cette *bitstring* avec la clé publique de l'autre partie. Les deux parties utilisent alors la valeur du `xor` de ces deux *bitstrings*, $e_I \oplus e_R$, comme nouvelle clé commune. Si l'on considère ce protocole, il est nécessaire de supposer que

3. Même proche de nous, voir par exemple `https://www.lemonde.fr/international/article/2022/01/12/une-application-anti-covid-utilisee-dans-une-enquete-policiere-cible-de-vives-critiques-en-allemagne_6109148_3210.html`.

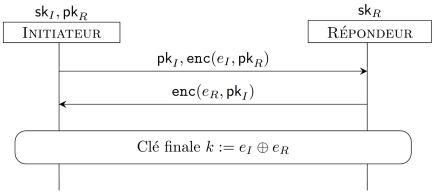

FIGURE 1. Exemple d'échange de clé

le chiffrement fournit certaines garanties pour montrer que la clé finale est effectivement secrète. Ainsi, une première étape pour fournir des garanties sur les systèmes de communication est donc de vérifier que ces protocoles sont sécurisés en supposant que les primitives le sont. Pour cela, il faut modéliser les protocoles, les propriétés de sécurité et les attaquants. C'est là qu'est la difficulté majeure, et l'une des différences avec la vérification classique de systèmes : on doit modéliser de manière réaliste une classe d'attaquants possibles, et montrer qu'aucun attaquant ne peut compromettre le système.

Modèles symbolique et calculatoire Aujourd'hui, deux modèles sont largement utilisés pour cela :

— le modèle symbolique, proposé par [3], où l'attaquant contrôle toutes les communications du réseau, les messages sont modélisés par des termes, et le pouvoir de l'attaquant pour manipuler les messages est capturé par une théorie équationnelle et des règles d'inférence ;

— le modèle calculatoire, initié par [4], où l'attaquant contrôle toutes les communications, peut être n'importe quelle machine de Turing s'exécutant en temps probabiliste polynomial, et les messages sont directement des suites de bits.

Quel que soit le modèle, les preuves peuvent être très complexes et très longues déjà sur des protocoles simples, et réaliser des preuves correctes à la main devient vite difficile, voire impossible. De nombreux outils ont alors été développés pour faire des preuves interactives ou automatiques. Les outils du modèle symbolique (TAMARIN, PROVERIF...) ont le plus haut niveau d'automatisation et permettent de vérifier des protocoles entiers avec des scénarios de compromission complexes. Les outils du modèle calculatoire (EASYCRYPT, CRYPTOVERIF, SQUIRREL...) sont quant à eux moins automatisés. Ils permettent seulement de vérifier des protocoles plus petits, avec des scénarios de compromissions moins complexes mais avec un attaquant plus réaliste. Il existe, de manière générale, un compromis entre le réalisme des modèles et la puissance de calcul de l'attaquant, et les approches symboliques et calculatoires sont complémentaires.

Pour répondre à notre problématique principale, nous pensons donc qu'il est nécessaire d'améliorer, de combiner et d'utiliser plusieurs outils de preuve pour vérifier et améliorer des standards de protocoles déployés à grande échelle.

Contributions de ma thèse

Dans ma thèse, nous avons progressé dans cette direction via plusieurs angles :

— **modularité** : nous avons proposé un cadre général de composition qui permet d'obtenir la preuve d'un protocole à partir de la preuve de ses composants ;

— **preuves assistées par ordinateur** : nous avons travaillé sur l'automatisation d'étapes de preuves bas niveau dans le modèle calculatoire à l'aide d'outils symboliques ou mathématiques ; et nous avons développé un nouveau prouveur interactif, SQUIRREL, permettant de fournir des garanties calculatoires sur la sécurité de protocoles complexes et qui a déjà servi à analyser des protocoles que les outils existants ne pouvaient pas considérer ;

— **mise en pratique** : nous avons effectué une étude de cas pratique, en poussant le modèle symbolique dans ses limites en considérant des attaquants aussi puissants que possibles dans le cadre de l'authentification multi-facteurs ; autour de 5 000 scénarios en 6 minutes ont été analysés via PROVERIF.

Notre résultat de composition à haut niveau

Pour simplifier la complexité des preuves et les rendre plus facilement réutilisables, de nombreux travaux ont essayé de permettre une analyse modulaire des protocoles. L'une des approches principales est l'*Universal Composability* [2], où l'on démontre qu'un composant est sécurisé quel que soit le contexte. L'inconvénient de cette approche est que devoir prouver la sécurité pour tout contexte implique souvent des hypothèses fortes sur les composants, hypothèses souvent non satisfaites en pratique. Le cadre d'application de cette ligne de recherche se retrouve ainsi limité lorsqu'il s'agit d'analyser des protocoles complexes du monde réel tel que SSH.

Une autre approche vise, au lieu de prouver la sécurité indépendamment du contexte, à prouver la sécurité de composants uniquement pour un ensemble restreint de contextes satisfaisant certaines conditions, et ensuite de prouver que le contexte concret satisfait ces hypothèses. Cependant, dans cette ligne de travail, les résultats existants ne s'appliquent qu'à un type de protocole précis, tels que les échanges de clés présentés précédemment, et ne sont pas génériques.

Pour répondre à ces limitations, notre objectif dans ce travail a été de développer un cadre de preuve générique et modulaire où l'on peut prouver qu'un composant est sécurisé dans un ensemble restreint de contextes satisfaisant un certain nombre de conditions, et il suffit de prouver que le contexte du cas d'application satisfait lesdites conditions pour obtenir la sécurité globale.

Nous souhaitions par ailleurs obtenir un résultat de composition qui soit applicable dans le cadre de la logique BC [1], une logique du premier ordre qui permet

de prouver la sécurité des protocoles en obtenant des garanties calculatoires. Si cette logique a rapidement montré son intérêt, et a, par exemple, servi à faire une analyse formelle de AKA à la main, l'une des faiblesses de cette logique était en effet de ne pouvoir considérer qu'un nombre fixe d'agents en parallèle ; faiblesse qu'un résultat de composition peut alors permettre de résoudre en réduisant la sécurité de plusieurs sessions en parallèle à la sécurité d'une unique session. Par ailleurs, cette logique étant la base de SQUIRREL, cela permet alors d'avoir un résultat de composition utilisable dans notre assistant de preuve.

Principale difficulté À un haut niveau d'abstraction, nous considérons des programmes distribués. Si ces programmes ne partagent aucune ressource, il est alors facile de montrer que l'exécution d'un programme n'affecte pas celle des autres, et que l'on peut alors faire des preuves sur chaque programme isolément puis en les combinant ensemble. Par contre, si deux programmes concurrents partagent un même état, cela n'est plus possible. Dans le cadre de la sécurité, si deux protocoles révèlent chacun une moitié distincte d'un secret, chaque programme en isolation ne révèle pas le secret mais le secret sera bien sûr compromis lorsque les deux protocoles s'exécutent en parallèle. Ainsi, la difficulté principale d'un résultat de composition est la capacité à gérer ces ressources que peuvent partager différents programmes.

Notre approche Soit P, Q deux programmes partageant une dépendance commune à une ressource s, par exemple un état ou une clé secrète. On souhaite démontrer que pour tout attaquant A sans accès à s, l'exécution en parallèle des programmes et de l'attaquant vérifie une propriété de sécurité φ qui ne parle que de P, cela dénoté $\forall A.\ (A||P||Q) \models \varphi$. On souhaiterait via un résultat de composition pouvoir faire une preuve qui ne considère pas toutes les exécutions possibles de Q, et se concentre sur P. Notre idée de base est de faire une preuve pour un attaquant A ayant un accès restreint à s lui permettant de simuler le comportement de Q. Si par exemple tous les accès à s dans Q sont de la forme $s := s+6$ ou $s := s+3$, alors il suffit d'augmenter le pouvoir de l'attaquant en lui donnant un moyen d'effectuer à volonté l'opération $s := s+3$ pour qu'il puisse simuler Q. Et une preuve de sécurité contre un tel attaquant est alors une preuve valide pour une exécution dans le contexte de Q.

Nous avons ainsi défini l'idée de simulation par oracle, où l'on donne à l'attaquant accès à un oracle $\mathcal{O}_s$, et on demande à l'attaquant de simuler Q à l'aide de cet oracle. Et ainsi, on peut prouver $\forall A.\ (A||P||Q) \models \varphi$ en démontrant que pour tout attaquant $A^{\mathcal{O}_s}$ ayant accès à l'oracle, $A^{\mathcal{O}_s}||P \models \varphi$. L'idée de la preuve étant que s'il existe un attaquant A et une exécution de P et Q en parallèle violant la propriété, alors il existe également un attaquant $A^{\mathcal{O}_s}$ qui va pouvoir simuler parfaitement cette exécution en interagissant uniquement avec P et donc violer la propriété.

Notre théorème principal établit que pour tout Q tel que la seule ressource commune avec P est s, si $\forall A^{\mathcal{O}_s}.A^{\mathcal{O}_s}||P \models \varphi$ et si Q est simulable grâce à $\mathcal{O}_s$, alors nous

avons que $\forall A.(A||P||Q) \models \varphi$. Nos résultats sont plus généraux, et permettent de couvrir la composition parallèle comme illustré ici, mais aussi la composition séquentielle et surtout la réplication, où l'on considère en parallèle un nombre non borné de copies d'un protocole.

Application à la preuve de sécurité L'approche de la logique BC a été « d'axiomatiser » ce que l'adversaire ne peut pas faire, ce qui a de nombreux avantages et permet de se détacher des détails des modèles d'attaquants et de dériver la propriété de sécurité de la formalisation du protocole et de ces axiomes. Cela permet d'obtenir des résultats dans le modèle calculatoire, dont l'approche est justement de définir ce que l'attaquant ne peut pas faire, e.g., qu'un problème est calculatoirement dur, plutôt que de définir l'attaquant par ce qu'il peut faire comme le fait le modèle symbolique. Le résultat de composition ci-dessus s'adapte alors parfaitement, car il suffit maintenant de considérer des axiomes qui sont corrects pour un adversaire ayant accès à l'oracle $\mathcal{O}_s$. Notre deuxième résultat théorique consiste à montrer concrètement que de tels axiomes peuvent être facilement dérivés des axiomes standards de la cryptographie, par exemple pour les propriétés d'intégrité des signatures digitales.

Enfin, nous avons montré des exemples concrets d'applications pour les protocoles d'échanges de clés. C'est un cas classique de composition, où par exemple après les opérations de figure 1, la nouvelle clé commune peut être utilisée par un autre protocole entre I et R qui est alors composé séquentiellement avec l'échange de clé. Plus spécifiquement nous avons pu, grâce à notre résultat, effectuer la première preuve de sécurité de SSH [5] avec des capacités de forwarding agent pour un nombre arbitraire de sessions.

Références

[1] Gergei Bana and Hubert Comon-Lundh. A Computationally Complete Symbolic Attacker for Equivalence Properties. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS'14)*, pages 609–620, Scottsdale, Arizona, USA, November 2014. ACM Press.

[2] Ran Canetti. *Universally Composable Security : A New Paradigm for Cryptographic Protocols*. 2000.

[3] D. Dolev and A. Chi-Chih Yao. On the security of public key protocols. In *22nd Annual IEEE Symposium on Foundations of Computer Science, FOCS 1981*, pages 350–357. IEEE Computer Society, 1981.

[4] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2) :270–299, 1984.

[5] Tatu Ylonen and Chris Lonvick. The Secure Shell (SSH) Transport Layer Protocol.