Secure Composition of Protocols*

Véronique Cortier

LORIA, CNRS, project Cassis, Nancy, France

Abstract. Security protocols are small distributed programs that are designed to ensure security over untrusted networks such as the Internet. They are notoriously dificult to design and flaws can be found several years after their publication and even their deployment. In particular, they are not securely composable in general: two protocols may be secure when analyzed separately but may cause harmful interactions to each other. We explore how tagging protocols allows to securely compose protocols.

Security protocols are small distributed programs that are designed to ensure security over untrusted networks such as the Internet. Examples of such programs are protocols for e-payment, pay-per-view, authentication or e-voting. They are notoriously dificult to design and flaws can be found several years after their publication and even their deployment.

A recognized way of ensuring a better security level is to analyse security protocols using formal methods, providing rigorous proofs of their security. In this vein, several decision procedures have been developed for checking security properties such as confidentiality or authenticity (e.g. [14,11,2,5,8,16]). These decision procedures have often yield tools that are able to automatically prove security or discovering flaws if any (e.g. [4,3,13,17,15,10]). While automatic tools are successful in analyzing protocols in isolation, they do not perform well when used for analyzing several protocols combined together. We summarize here two approaches (previously presented in [9] and [7]) that show how to securely compose protocols. These results allow to analyze the security of protocols component by component and then deduce directly the security of the combined protocol.

Parallel composition

Many protocols are executed simultaneously over the Internet and they may share some datas such as keys. This is in particular the case of protocols when several versions of the same protocol may be used at the same time (because not everybody uses the most up-to-date version), when several modes are allowed like in IKE, or when protocols make use of public keys. Even if a protocol has

* The research leading to these results was performed as part of the ProSecure project which is funding by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 258865.

S. Mödersheim and C. Palamidessi (Eds.): TOSCA 2011, LNCS 6993, pp. 29–32, 2012.

been proved secure against an active attacker that can read, block and create messages, there are absolutely no guarantees that it remains secure when other protocols (possibly variants of it) are executed in parallel in case they share some secret information. Examples of unwanted interactions between protocols can be found in e.g. [12]. To illustrate this discussion, we describe here the toy protocol proposed in [9].

$$P_1: \quad A \to B : \{s\}_{\mathsf{pub}(B)} \qquad\qquad P_2: \quad \begin{array}{l} A \to B : \{N_a\}_{\mathsf{pub}(B)} \\ B \to A : N_a \end{array}$$

In protocol P_1, the agent A simply sends a secret s encrypted under B's public key. In protocol P_2, the agent sends some fresh nonce to B encrypted under B's public key. The agent B acknowledges A's message by forwarding A's nonce. While P_1 executed alone easily guarantees the secrecy of s, even against active adversaries, the secrecy of s is no more guaranteed when the protocol P_2 is executed. Indeed, an adversary may use the protocol P_2 as an oracle to decrypt any message.

A way to avoid such a bad interaction is to *tag* messages, as proposed e.g. in [1,12,6,5]. Tagging messages consist in adding a protocol identifier in each cyphertext. For example, tagging P_1 and P_2 would result in the two following protocols.

$$P_1': \quad A \to B : \{1, s\}_{\mathsf{pub}(B)} \qquad\qquad P_2': \quad \begin{array}{l} A \to B : \{2, N_a\}_{\mathsf{pub}(B)} \\ B \to A : N_a \end{array}$$

The main result of [9] demonstrates that such tagged protocols can be safely executed simultaneously, provided that the shared information are either public or are only used for encryption and decryption.

Secure refinement

The situation might be even more complex when protocols are interleaved, one protocol establishing data for the other one and conversely. For example, some protocols run sub-protocols e.g. to establish confidential or authenticated channels. Moreover, protocols usually assume pre-established keys such as long-term symmetric keys or public keys associated with their legitimate owners. Such datas are actually established themselves running other protocols such as key-establishment protocols. Again, there is absolutely no guarantee that a protocol proved secure assuming pre-established keys remains secure independently of the way long-term keys are established. For example, consider the two following protocols.

$$Q_1: \quad A \to B : \{k_2\}_{K_1} \qquad\qquad Q_2: \quad A \to B : k_2, \{s\}_{k_1}$$

Protocol Q_1 establishes a key k_1 between A and B using some long-term key k_2. Protocol Q_2 assumes two pre-established keys k_1 and k_2, reveals k_2 and uses k_1 for transmitting a secret s. The protocol Q_2 alone clearly guarantees the

confidentiality of s. However, if Q_1 is used for establishing k_1 in Q_2 then Q_2 is no longer secure. The main result of [7] shows how to securely compose protocols by using distinct cryptographic primitives in each protocol (e.g. Diffie-Helmann for establishing keys in one protocol and standard encryption and signature schemes in the second protocol). If the same primitive needs to be used (e.g. encryption) then tagging protocols again allows to securely combine protocols together.

Acknowledgment. These results have been obtained in joint work with Stéphanie Delaune [9] for and joint work with Stefan Ciobaca for [7].

References

1. Abadi, M., Needham, R.M.: Prudent engineering practice for cryptographic protocols. IEEE Trans. Software Eng. 22(1), 6–15 (1996)
2. Amadio, R., Charatonik, W.: On name generation and set-based analysis in the Dolev-Yao model. In: Brim, L., Jančar, P., Křetínský, M., Kučera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 499–514. Springer, Heidelberg (2002)
3. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heám, P., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The Avispa tool for the automated validation of internet security protocols and ap plications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
4. Blanchet, B.: An efficient cryptographic protocol verifier based on Prolog rules. In: Proc. 14th Computer Security Foundations Workshop (CSFW 2001), pp. 82–96. IEEE Comp. Soc. Press, Los Alamitos (2001)
5. Blanchet, B., Podelski, A.: Verification of cryptographic protocols: Tagging enforces termination. In: Gordon, A.D. (ed.) FOSSACS 2003. LNCS, vol. 2620, pp. 136–152. Springer, Heidelberg (2003)
6. Canetti, R., Meadows, C., Syverson, P.F.: Environmental requirements for authentication protocols. In: Okada, M., Babu, C. S., Scedrov, A., Tokuda, H. (eds.) ISSS 2002. LNCS, vol. 2609, pp. 339–355. Springer, Heidelberg (2003)
7. Ciobâcă, Ş., Cortier, V.: Protocol composition for arbitrary primitives. In: Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF 2010), Edinburgh, Scotland, UK, pp. 322–336. IEEE Computer Society Press, Los Alamitos (July 2010)
8. Comon-Lundh, H., Cortier, V.: New decidability results for fragments of first-order logic and application to cryptographic protocols. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 148–164. Springer, Heidelberg (2003)
9. Cortier, V., Delaitre, J., Delaune, S.: Safely composing security protocols. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 352–363. Springer, Heidelberg (2007)
10. Cremers, C.: Scyther - Semantics and Verification of Security Protocols. Ph.D. dissertation, Eindhoven University of Technology (2006)
11. Durgin, N., Lincoln, P., Mitchell, J., Scedrov, A.: Undecidability of bounded security protocols. In: Proc. of the Workshop on Formal Methods and Security Protocols (1999)

12. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1998)
13. Lowe, G.: Casper: A compiler for the analysis of security protocols. In: Proc. 10th Computer Security Foundations Workshop (CSFW 1997). IEEE Comp. Soc. Press, Los Alamitos (1997)
14. Rusinowitch, M., Turuani, M.: Protocol insecurity with finite number of sessions and composed keys is NP-complete. Theoretical Computer Science 299, 451–475 (2003)
15. Schneider, S.: Security properties and CSP. In: Proc. of the Symposium on Security and Privacy, Oakland, pp. 174–187. IEEE Computer Society Press, Los Alamitos (1996)
16. Seidl, H., Verma, K.N.: Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 79–94. Springer, Heidelberg (2005)
17. Song, D.X.: Athena: A new efficient automatic checker for security protocol analysis. In: Proc. 12th Computer Security Foundations Workshop (CSFW 1999), Mordano, Italy. IEEE Computer Society Press, Los Alamitos (June 1999)