Attack-preserving Security Protocol Transformations

Binh Thanh Nguyen, David Basin, and Christoph Sprenger

Institute of Information Security, ETH Zurich, Switzerland
{thannguy,basin,sprenger}@inf.ethz.ch

Abstract. The rigorous incremental development of security protocols has so far received much less attention than protocol verification techniques. In this work, we study security protocol transformations. These can serve both for simplifying protocols before verification and, in the other direction, for developing protocols by stepwise refinement of simple abstract protocols into complex concrete ones. The transformations preserve attacks on a class of security properties. Our work aims to improve our understanding of modifications of existing protocols and to enable the systematic development of entire families of new protocols. This complements existing work on post-hoc protocol verification.

1 Introduction

It is well-known that security protocols are notoriously hard to get right. This motivates the use of formal methods for their design and development. In the last decade, we have witnessed substantial progress in the formal verification of security protocols. However, methods for developing security protocols have received much less attention and protocol development remains more an art than a science.

Sprenger and Basin [14,15] have recently proposed a hierarchical development method for security protocols based on stepwise refinement that spans several levels of abstraction. Each development starts from abstract models of security properties and proceeds down to cryptographic protocols secure against a Dolev-Yao intruder. The development process traverses intermediate levels of abstraction based on message-less protocols and communication channels with authenticity and confidentiality properties. Security properties, once proved for a given model, are preserved by further refinements. They have applied their method to develop families of authentication and key transport protocols. However, developers may not be familiar with these abstractions and the underlying refinement framework. They are more familiar with cryptographic messages and transforming these messages to create new protocols from existing ones.

This motivates our study of refinements in terms of protocol transformations at the level of cryptographic messages. In particular, we are interested in protocol transformations that preserve attacks against a given set of security properties from concrete protocols to abstract ones (or, equivalently, the satisfaction of such properties in the reverse direction). Such transformations can serve the systematic development of individual protocols and entire families of protocols. Moreover, they can be applied to modify or compare existing protocols and understand their differences. The modification of security protocols is particularly error-prone (see, e.g., [2]). Security protocol standards

constitute another relevant application field for protocol transformations, since they typically comprise numerous protocol variants and options.

Security protocol transformations can also be considered as abstractions (i.e., from concrete to abstract). Hui and Lowe [10] define several kinds of attack-preserving transformations with the aim of simplifying protocols so that they can be effectively verified using model checking. They define criteria for the preservation of secrecy and authentication properties, and prove for each kind of transformation that it satisfies these criteria.

Datta et al. [6,5] use protocol templates with messages containing function variables to specify and prove properties of protocol classes. Their notion of refinement is based on instantiating function variables and discharging the associated assumptions. Pavlovic et al. [13,3] similarly refine protocols by transforming messages and propose specialized formalisms for establishing secrecy and authentication properties. Unfortunately, their approach lacks a formal semantics.

Guttman [9,8] studies the preservation of security properties by a rich class of protocol transformations in the strand space model. His approach to property preservation is based on the simulation of protocol analysis steps instead of execution steps. Each analysis step explains the origin of a received message. However, he does not provide syntactic conditions for the transformations to preserve security properties.

The objective of our work is to develop a comprehensive theory of protocol transformations covering a wide range of protocols and security properties. Our starting point is Hui and Lowe's work [10]. They work in a restricted protocol model with typed messages and atomic keys and show their results for ground messages. However, in order to transform protocol descriptions, we have to consider messages with variables and justify that a transformed attack is indeed an execution of the abstract protocol. They only discuss this important point briefly and informally. We plan to address these issues and obtain preservation results for relevant classes of security protocols (such as those based on convergent subterm theories [1]) and expressive property specification languages (such as PS-LTL [4] or the language proposed in [11]). We aim to cover a large class of protocol transformations including those described in the examples in [3,6,5].

We intend the following contributions. We want to significantly extend the scope of existing work in terms of expressiveness of the protocol specifications, the protocol transformations, and the preserved properties. Our work will provide a sound formal underpinning to protocol transformations, which can serve as a foundation for rigorous security protocol development, modifications, and comparisons of existing protocols.

2 A motivating example

We present the development of a key transport protocol as a motivating example. We state the protocols in standard Alice&Bob notation and describe each refinement step as a protocol transformation.

Consider a key transport protocol P_1, where a key server S generates and distributes a session key K_{AB} to an initiator A and a responder B.

M1.1. $A \to S : A, B$
M1.2. $S \to A : \{B, K_{AB}\}_{K_{AS}}$
M1.3. $S \to B : \{A, K_{AB}\}_{K_{BS}}$

In order to prevent replays and guarantee the recentness of K_{AB}, we refine this protocol into P_2 by adding a nonce and a timestamp to P_1.

$$\begin{aligned}
&\text{M2.1.}\ \ A \to S : A, B, N_A\\
&\text{M2.2.}\ \ S \to A : \{B, T_S, N_A, K_{AB}\}_{K_{AS}}\\
&\text{M2.3.}\ \ S \to B : \{A, T_S, K_{AB}\}_{K_{BS}}
\end{aligned}$$

Next, we obtain P_3 by refining the flow of protocol messages: the server now appends B's message in M2.3 to A's in M2.2, which A then forwards to B.

$$\begin{aligned}
&\text{M3.1.}\ \ A \to S : A, B, N_A\\
&\text{M3.2.}\ \ S \to A : \{B, T_S, N_A, K_{AB}\}_{K_{AS}}, \{A, T_S, K_{AB}\}_{K_{BS}}\\
&\text{M3.3.}\ \ A \to B : \{A, T_S, K_{AB}\}_{K_{BS}}
\end{aligned}$$

In P_3, B cannot be sure that A has received the key K_{AB}. We refine P_3 into P_4 by having the server encrypt B's message inside A's, which allows A to authenticate B on K_{AB}.

$$\begin{aligned}
&\text{M4.1.}\ \ A \to S : A, B, N_A\\
&\text{M4.2.}\ \ S \to A : \{B, T_S, N_A, K_{AB}, \{A, T_S, K_{AB}\}_{K_{BS}}\}_{K_{AS}}\\
&\text{M4.3.}\ \ A \to B : \{A, T_S, K_{AB}\}_{K_{BS}}
\end{aligned}$$

Protocol P_4 is a basic form of the Kerberos IV protocol (without authenticators). We have started from a simple initial protocol guaranteeing only the secrecy of the session key. We have then used refinement to add several features to this protocol in order to obtain key freshness, recentness, and authentication properties.

For additional examples of protocol developments, we refer the reader to [3,6,5].

3 Approach and current work

Security protocol model We briefly summarize our security protocol model, which is based on [11]. The model is parametrized by a message term algebra over a given signature Σ and a set of variables $\mathcal{V}$. Constants model nonces, keys, time stamps, and agents. Function symbols typically include hashes $h(t)$, pairs $\langle t, u\rangle$, and encryptions $\{t\}_u$. Let $\mathcal{T}$ be the set of all terms over Σ and $\mathcal{V}$. The terms may be quotiented by an equational theory, e.g., to model the commutativity of exponentiation for a Diffie-Hellman protocol. As is standard, we model a Dolev-Yao intruder [7] with full control over the network using a deduction system. Its judgements have the form $T \vdash u$, meaning that the intruder can derive the term u from a finite set of terms T. Encryption is perfect, i.e., the intruder can only decrypt with the intended key.

We specify protocols as finite sets of roles instead of the informal Alice&Bob notation from Section 2. Each role $R \in \mathit{Role}$ is a sequence of send and receive events of the form $\mathsf{snd}(t)$ or $\mathsf{rcv}(t)$ for a term t. The semantics of a protocol is a transition system with states of the form $s = (tr, th, \sigma)$, where tr is a trace consisting of a sequence of pairs of thread identifiers and events, $th : TID \rightharpoonup \mathit{Role}$ is a thread pool, and $\sigma = \{\sigma_i \mid i \in \mathit{dom}(th)\}$ is a family of ground substitutions σ_i, one for each thread i. The transitions are defined by an operational semantics with rules for sending and receiving messages. The receive rule includes a premise requiring that the received message is deducible by the intruder from his initial knowledge and the sent messages. We write $\mathcal{R}(P)$ for the set of reachable states of the protocol P.

Protocol transformations Our protocol transformations are functions $f\colon \mathcal{T} \to \mathcal{T}$ on terms, which we lift to events, roles, protocols, traces, and states. We consider a class of *nice* transformations, which includes the following operations on messages:

1. remove encryptions and hashes,
2. remove fields from an encrypted message,
3. pull fields outside of an encryption,
4. split encryption into several ones, and
5. project pairs (under certain conditions) and reorder pairs.

These protocol transformations simplify messages (and hence protocols) and can therefore be understood as abstractions. However, the same transformations can be used for protocol refinements, which proceed in the opposite direction, from abstract to concrete. For example, in Section 2, the refinement of P_1 into P_2 uses transformations of the second type, and the one from P_3 into P_4 uses a transformation of the third type.

So far we do not cover structural transformation of protocol like the message relaying transformation (cf. the refinement of P_2 to P_3), but we plan to do so in the future.

Property specification language We consider a property specification language with formulas of the following shape.

$$\phi = \forall i_1, \ldots, i_m. \bigwedge_{A \in \Gamma} A \;\Rightarrow\; \exists j_1, \ldots, j_n. \bigwedge_{B \in \Delta} B \tag{1}$$

The quantifiers range over thread identifiers and Δ, Γ are sets of atomic predicates. These predicates include $\mathit{learns}(m)$ for expressing intruder knowledge in secrecy properties, and event orderings $e \prec e'$ and equations $m = m'$ for authentication properties. To achieve attack preservation, the $\mathit{learns}(m)$ is only allowed to occur in Γ. A state $s = (tr, th, \sigma)$ that does not satisfy a property ϕ, written $s \not\models \phi$, is called an attack on ϕ.

Attack preservation Suppose we are given a class of security protocols, properties, and transformations such as those sketched above. The main result we want to achieve is the preservation of attacks on a property ϕ of protocol P to attacks on the transformed protocol $f(P)$ and property $f(\phi)$. We formalize this property as follows.

$$\begin{aligned} &\forall (tr, th, \sigma) \in \mathcal{R}(P).\ (tr, th, \sigma) \not\models \phi \\ &\quad \Rightarrow (f(tr), f(th), f(\sigma)) \in \mathcal{R}(f(P)) \wedge (f(tr), f(th), f(\sigma)) \not\models f(\phi) \end{aligned} \tag{2}$$

We decompose the proof of such results into two parts: the preservation of (i) executability (first conjunct) and (ii) attacks (second conjunct).

Executability The proof that for each reachable state (tr, th, σ) of P the transformed state $(f(tr), f(th), f(\sigma))$ is reachable in $f(P)$ is based on the following deducibility preservation result.

$$T\theta \vdash u\theta \Rightarrow f(T)f(\theta) \vdash f(u)f(\theta) \tag{3}$$

This follows from two simpler properties. The first one is a simpler version of (3).

$$T \vdash u \;\Rightarrow\; f(T) \vdash f(u) \tag{4}$$

The second one requires that f satisfies the following substitution property, that is, for all terms t and substitutions θ,

$$f(t\theta) = f(t)f(\theta) \tag{5}$$

In particular, since the operational semantics of receive events requires the deducibility of the received message from previously sent messages, we can use (3) to show that each receive event of P can be simulated by a corresponding receive event in $f(P)$.

Attack preservation Since secrecy is expressed in terms of deducibility of messages, we obtain the preservation of secrecy for free from the above. For other properties, like those expressible in the language sketched above, a separate proof is needed.

We have proved property (4) for all nice transformations. However, the substitution property (5) turns out to be quite restrictive. It rules out transformations that look more than one level into the term structure (such as, e.g., for splitting an encryption). Our initial solution restricts the set of substitutions to *simple* ones, whose range contains no composed terms. This set covers typed substitutions, which are (implicitly) used in [10]. We have proved (5) (hence *executability*) and *attack preservation* for this restricted setting and a subclass of nice transformations specified by pattern matching. Unfortunately, this solution rules out untyped variables such as those required for forwarding messages (cf. Section 2).

4 Planned work and conclusions

Generalizing the results An alternative solution is based on the observation that executability depends on constraints $T \vdash u$ where the terms in T stem from send events and u from a receive event. Therefore, a restricted form of (5) where t ranges over the set of terms in the protocol roles suffices for executability. Since this form of (5) is protocol-dependent, we cannot use induction to establish it. Instead, we need to formulate criteria to check that a protocol has this property. For attack preservation, the substitution property must also hold for the terms occurring the properties we are interested in.

A different approach could replace the substitution $f(\theta)$ in (3) by some θ'. The construction of such a θ' would require a stronger proof technique, possibly based on symbolic constraint reduction [12]. This approach produces non-ground substitutions as solutions of constraint systems. Therefore, we would have some freedom to derive different ground substitutions θ'.

Outlook on future work In a longer-term perspective, we plan to extend the scope of transformations in several directions. First, we want to cover structural transformations, which not only modify messages, but also events and roles (e.g., relaying messages; splitting, merging, and deleting events). Second, we would like to cover a larger class of protocols, in particular, by including equational theories (e.g., Diffie-Hellman exponentiation, convergent subterm theories [1]). Third, we intend to extend the property language to include additional properties such as forward secrecy and also consider stronger adversary models (e.g., compromising session keys and local states). Finally, we also plan to implement a tool that supports the definition and application of protocol transformations and the guarantee of their soundness.

Conclusions In this work, we study attack-preserving security protocol transformations. These can be used for the abstraction, the refinement, and the comparison of protocols. Therefore, we consider this technique as a useful complement to verification.

So far, we have defined a subclass of transformations and proved the preservation of attacks with respect to a particular security property language. We have discussed the problems that we have encountered and proposed possible solutions. We have also sketched our plans for future work.

Acknowledgements This work is partially supported by the EU FP7-ICT-2009.1.4 Project No. 256980, NESSoS: Network of Excellence on Engineering Secure Future Internet Software Services and Systems.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006.
2. A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps. In V. Shmatikov, editor, *FMSE*, pages 1–10. ACM, 2008.
3. I. Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations*, pages 48–61, Washington, DC, USA, 2005.
4. R. Corin, S. Etalle, and A. Saptawijaya. A logic for constraint-based security protocol analysis. In *IEEE Symposium on Security and Privacy*, pages 155–168, 2006.
5. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 30–, 2004.
6. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositionl logic for security protocols. *Journal of Computer Security*, 13:423–482, 2005.
7. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
8. J. D. Guttman. Transformations between cryptographic protocols. In P. Degano and L. Viganó, editors, *ARSPA-WITS*, volume 5511 of *LNCS*, pages 107–123. Springer, 2009.
9. J. D. Guttman. Security goals and protocol transformations. In *Theory of Security and Applications (TOSCA), an ETAPS associated event*, volume 6993 of *LNCS*. Springer, 2011.
10. M. L. Hui and G. Lowe. Fault-preserving simplifying transformations for security protocols. *Journal of Computer Security*, 9(1/2):3–46, 2001.
11. S. Meier, C. J. F. Cremers, and D. A. Basin. Strong invariants for the efficient construction of machine-checked protocol security proofs. In *Proc. 23th IEEE Computer Security Foundations Symposium (CSF)*, pages 231–245, 2010.
12. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.
13. D. Pavlovic and C. Meadows. Deriving secrecy in key establishment protocols. In *Proc. 11th European Symposium on Research in Computer Security (ESORICS)*, pages 384–403, 2006.
14. C. Sprenger and D. Basin. Developing security protocols by refinement. In *Proc. 17th ACM Conference on Computer and Communications Security (CCS)*, pages 361–374, 2010.
15. C. Sprenger and D. Basin. Refining key establishment. Technical Report 736, Computer Science Department, ETH Zurich, Sept. 2011.