Available online at www.sciencedirect.com

SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 142 (2006) 99–110

www.elsevier.com/locate/entcs

Compositionality of Security Protocols: A Research Agenda

Cas Cremers[1]

Department of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands

Abstract

The application of formal methods to security protocol analysis has been extensively researched during the last 25 years. Several formalisms and (semi-)automatic tools for the verification of security protocols have been developed. However, their applicability is limited to relatively small protocols that run in isolation. Many of the protocols that are in use today cannot be verified using these methods. One of the main reasons for this is that these protocols are composed of several sub-protocols. Such a composition of protocols is not addressed in the majority of formalisms. In this paper we identify a number of issues that are relevant to applying formal methods to the problem of security protocol composition. Additionally, we describe what research needs to be done to meet this challenge.

Keywords: Formal methods, Security protocol analysis, Compositionality, Security properties.

1 Introduction

Within the areas of Internet, e-commerce and smart card applications, the use of cryptographic primitives is by now standard practice. In the last few decades much research has covered the mathematical issues involved in such cryptographic primitives, and many of these are by now well understood. However, using such well-understood cryptographic primitives to implement a protocol does not guarantee that the protocol itself satisfies a security requirement (See e.g. [10]). This observation has guided a branch of research on security protocols where the cryptographic primitives are assumed to be

[1] Email: ccremers@win.tue.nl

doi:10.1016/j.entcs.2004.12.047

perfect, and a black-box view of cryptography is used (cf. [8]). Using formal models, it is possible to prove that a security protocol is correct (see [16,6]) under some assumptions. Furthermore, there are various tools that can find attacks on security protocols such as [4,11,13].

Although much advances have been made, the protocols that have been investigated are still modest in size. The available proof methods, as well as the tools, do not scale well. In the same vein, the formal models assume what we call a *closed world*. This means that the agents in the system do not exhibit any other behaviour than is specified in the protocol.

This closed world assumption does not hold for most applications. For example, agents on the Internet are usually capable of many types of behaviour, and use multiple protocols concurrently.

But even if the closed world assumption does not hold, folklore and common sense indicate that as long as parallel protocols share no common data, no harm is done. Many designs rely on this hypothesis and use different keys for different protocols.

On the other hand, in many contexts it is desirable to have keys that are shared among protocols. When designing protocols for smart card applications, resources such as memory and bandwidth are limited and thus it makes sense to re-use key material. Even when designing a complex system where such resources are sufficiently available, it can be very expensive to introduce a key infrastructure for each sub-protocol.

We consider verification of a single protocol under the closed world assumption to be often unrealistic, as well as undesirable for the design of complex systems involving security protocols. Therefore, we set out to enlarge this closed world, and improve the scalability of the existing formal methods. Furthermore, we suggest a new level of abstraction for the composition of security protocols. We identify a number of open issues, to be addressed by future work on security protocols.

The remainder of this paper is organised as follows. In Section 2, we investigate the closed world assumption and give some examples to indicate the problem areas. Then, in Section 3, we discuss how to model a larger closed world, and indicate requirements for the semantics of such a world. We sketch a new level of abstraction specifically tailored for the problem in Section 4. Some related work is discussed in Section 5, and we close off in Section 6 with concluding remarks and a number of open research questions.

2 Composition of Security Protocols

As stated in the introduction, there exist several formal methods to establish that a security protocol achieves some security goals. One of the assumptions made by these methods is the *closed world assumption*, i.e. that the agents involved exhibit no other behaviour than that which is described by the protocol. This assumption, combined with an explicit intruder model (such as Dolev-Yao, see [8]), enables us to describe all possible states of the system which is the composition of the agents and the intruder. The fact that every possible behaviour of this system is modeled, allows some properties of the system to be proven.

If we execute several (provably correct) protocols in the same environment, it can be the case that the goals are not achieved anymore. An intruder can use parts of a protocol to generate messages that he could not construct otherwise. When multiple protocols are executed concurrently, the number of messages that the intruder can forge increases exponentially. This can result in new ways of attacking a protocol. An example of such an attack on an early version of SSL is described in [1]. Another attack on a composed protocol can be found in [15].

It is possible to verify such protocol collections by modeling them as a single larger protocol, but the resulting protocol is typically very large, and therefore outside the scope of current methods.

The upshot of this is, that there are small protocols known that provably achieve security goals such as secrecy or authentication, under some assumptions such as the closed world assumption. In isolation these protocols are not very useful. In real world applications they will be used either concurrently or appended to other protocols, thereby invalidating the closed world assumption of the proof. Consequently, the resulting system will not necessarily meet the security goals.

The current practice is that large or composed protocols are not formally verified. The verification tools can verify small protocols, but there are no guidelines as how to combine them into larger protocols without introducing new attacks.

We would like to remedy this situation, and make formal methods applicable to real life systems. To facilitate this, and make the formal methods usable for larger systems, a number of challenges need to be addressed. In the next two sections we sketch two possible approaches to the problem.

3 Improving the scalability

A major drawback of the formal approaches to security protocol analysis is their lack of scalability. We find that the size of security protocols that can be analysed, is still fairly small when compared to the protocols that are used in e.g. Internet protocols such as SSL. Many of the protocols that are in use today consist of a collection of sub-protocols, composed in parallel. In general such a larger, composed protocol cannot be verified using current formal methods.

We believe that many of the current methods do not scale well for such protocols, due to three main reasons.

(i) State space explosion. If a protocol is extended with a number of messages, the state space of the system grows exponentially. Note that this problem is not specific to security protocol analysis.

(ii) Lack of support in the formal models to capture sub-protocol composition.

(iii) Lack of support in the formal models to express security goals for sub-protocols.

Some attempts have already been made to address these problems. The state space explosion problem is not specific to security protocols, and occurs in many models of concurrent processes. It can sometimes be remedied using symbolic representation, or by providing invariants or other state space restrictions for specific instances of the model, but these techniques typically require (creative) user input. However, dealing with this is not within the scope of this article and we refer the reader to e.g. [2].

The second problem, the lack of support of (sub)protocol composition, has been addressed within the Strand Spaces framework from [16]. Within the original Strand Spaces framework it is not possible to model parallel protocols. This has lead to a minor modification of the Strand Spaces model, which is called the Mixed Strand Spaces model (see [15]). In this model, notions of primary and secondary strands are introduced, which allows a protocol to be analysed in the presence of other protocols. However, in this mixed model the third problem of specifying subgoals is left to the user modeling the system. Great care has to be taken when composing two sub-protocols, as the resulting set of goals might not be consistent.

A more recent formal model which can handle the parallel composition of protocols and goals, is the Operational Semantics of Security Protocols as defined in [6]. In this model, the notion of a protocol role is considered the basic unit of modeling. The model of a simple protocol consists of a collection of roles, and within these semantics the parallel composition of

protocols is captured by simply adding more roles to the system. Security goals are modeled as events in the protocol, representing local claims, and can be composed without any problem.

Much work remains to be done. We identify a number of hurdles that still need to be taken.

3.1 Determination of bound on runs

In general, tool based verification of security protocols assumes a bound of the number of runs, or protocol role instances, that are involved in an attack. Such a bound is used to restrict the state space of the problem and make automated verification feasible. Sometimes this bound is explicitly determined using hand proofs as in [10], but in most cases a low bound is simply assumed. This implies that most tools can prove that there is attack, but they cannot prove that a protocol is secure beyond the chosen bound. The reason for this is that there are no proofs known that yield a usable theoretical bound for any protocol.

Until now, verification using tools was based on some rule of thumb (e.g. two arbitrary instances of each role) or experience (attacks in literature involve three or four runs at most). When we are verifying composed protocols, such a bound becomes relevant again, as even having two instances of each role might lead to a state space that is far too large to explore. If possible, we are looking for an automatic procedure that, given a specific protocol and security goals, can determine an upper bound on the runs involved in an attack.

3.2 Support for protocol composition

The majority of security protocol models, do not support composed protocol analysis. A first step towards verification of these protocols is to adapt the semantics of the formal models. It might be the case that some models cannot easily be adapted for such a task, so research is required.

The adaption of the semantics of the models is also required for tooling. If a certain model does not support protocol composition, then neither will the tools based on the model. This is the case with the e.g. the Casper/FDR toolset (see [11]). The Athena tool (see [13]) supports parallel composition of protocols, but requires an expert user to provide suitable input. The Scyther tool (see [4]) supports composition in an intuitive way, by simply concatenating the protocol descriptions.

3.3 Support for security requirements composition

There are two main reasons why the composition of security requirements is non-trivial.

The first reason is that security properties were usually designed with just simple protocols in mind. Many security properties are defined as two-party properties, e.g. Alice authenticates Bob, Alice shares a key with Bob. Such properties can introduce ambiguities in a composed protocol context.

As an example, consider two parallel protocols $P1$ and $P2$ that both use the same base authentication protocol. Suppose Alice starts an instance of protocol $P1$. Because the base protocols are identical, she ends up talking to a responder instance of Bob. She can prove she is talking to Bob, but she cannot be sure which protocol he is using. Suppose protocol $P1$ uses the random values from the base protocol to create a session key, whereas $P2$ uses these to create a publicly known session identifier. Alice creates a session key from the value, but Bob, who is executing protocol $P2$, actually makes the random values public. Thus, the different semantics for the random values can cause serious security problems. Note that in both cases Alice and Bob *agree* over the random values that they used.

A strong authentication property that can be safely composed is *synchronisation* as defined in [7]. However, other properties need to be revisited as well in the light of requirement composition.

A second reason why such a composition is non-trivial, is because many formalisms consider the security requirement to be a seperate entity from the protocol specification, as in e.g. the strand spaces model. On one hand we have a protocol specification, and on the other hand there are seperate logical formulas for the requirements. When we compose two protocols, we also need to compose these logical formulas. Whether such a composition yields a valid security requirement is an open question.

An alternative approach is taken in our Operational Semantics for Security Protocols, where security property claims are modeled as events in the protocol. Thus, the requirements are integrated into the protocol specification. If two protocols are composed, the requirements are also composed. The security requirements have been modeled with the possibility of parallel protocols in mind.

3.4 Compositionality of proofs

Although the Mixed Strand Spaces and the Operational Semantics mentioned both look promising, their usefulness is limited by the fact that in general protocol proofs are not compositional. Thus, for the composition of two prov-

ably correct protocols, a new proof has to be constructed. In some cases we can avoid this: A result from the Mixed Strand Spaces model states that if two protocols are sufficiently dissimilar (see [9]), the correctness of the security goal of one protocol is independent of the presence of the other protocol. Such results are useful for the design of protocols. However, we need more of these types of results in order to reason about the composition of protocols.

3.5 Specifications of the context of security protocols

When a security protocol is formally specified, it is common to specify its requirements when it runs in isolation. This is connected to the way verification proceeds. However, when designing a protocol we can already indicate under which assumptions it will remain correct when composed with other protocols.

As an example, consider a protocol intended to generate and distribute a secret session key, which will be used to authenticate a sequence of messages, and keep their contents secret. After the key is known to the involved agents, the formal protocol typically ends. The protocol implicitly assumes that the session key is then used in a correct way, e.g. such that it remains secret. This is not expressed in the key exchange protocol, which has only the goal of establishing the key. However, such a protocol only makes sense if we actually use the session key in some other protocol, and it must therefore always be kept secret.

Most of the present protocols are not meant to be executed in isolation. Furthermore, many design decisions are made because these protocols are meant to be composed with other protocols. Thus, it is important to make these assumptions on the environment explicit.

However, current protocol specification formalisms are not suited to express this. We need ways of expressing the exact requirements of a protocol. For the example, we need to enforce some rules on how the session key is to be used. Furthermore, the protocol might make assumptions about an existing public/private key infrastructure, and introduce some proof obligations for the resulting composed system. One important research question would be to find concise ways of expressing these requirements.

Once such requirements for the context of protocols can be specified, we can start to consider these small protocols as *building blocks* for larger composed protocols. If we fix a set of small protocols to serve as building blocks, we can consider them as primitives for another level of abstraction. We explore this option in the next section.

4 Yet another level of abstraction

The standard Dolev-Yao model of black-box security protocols depends on cryptographic primitives such as encryption and signing. One reason for the research into black-box security protocols is that even if perfect cryptographic primitives exist, a protocol can still be attacked, as is shown by the usual example of the Needham-Schroeder protocol in [10].

The security protocol analysis works at a different level of abstraction than the cryptographic primitives. At the protocol level, we have security goals such as secrecy and authentication, that differ from their counterparts on the cryptographic level. Furthermore, new concepts are introduced such as injectivity.

This layer of abstraction has enabled formal analysis and automatic verification of security protocols. However, it seems that parallel protocol composition is stretching the limits of the protocol model, and one way out would be another level of abstraction.

We propose to introduce a new layer of abstraction, tailored specifically for the composition of security protocols. Such an abstraction will require security goals to be formulated at a protocol composition level. Corresponding to this, a new set of primitives, consisting of e.g. security protocols and message encapsulation, will have to be formulated.

Note that such an abstraction layer will not have the intention to assist the verification of existing protocols, rather it will be used for construction and design of new protocols. Therefore, the fundamental issue will be choosing an appropriate set of primitives and related concepts.

As an example, consider an authentication protocol. Such a protocol achieves e.g. agreement or synchronisation (as defined in [12] and [7]). In practice, and in a protocol composition setting, we would like to extend such an authentication protocol with another message exchange, such that the new messages are also authenticated. We believe that such a message extension is an integral part of the authentication protocol. Therefore, a composition primitive for authentication would function as an encapsulation for a sequence of message exchanges. Given a message exchange, we would apply such an authentication primitive, and the resulting protocol would satisfy synchronisation.

For the protocols in the standard libraries such as the Security Protocols Open Repository in [14] we have found Message Sequence Charts sufficiently expressive. However, for compositional issues such as a protocol that allows an agent to choose between sub-protocols, we expect that a graphical formalism such as e.g. High Level Message Sequence Charts will be required.

4.1 Develop concepts and primitives

At the protocol composition layer, we expect security goals such as secrecy to be straightforward extensions of their protocol counterparts. For goals such as authentication, privacy, and non-repudiation, we expect that significant modifications will have to be made to the concepts, in order to apply to protocol composition.

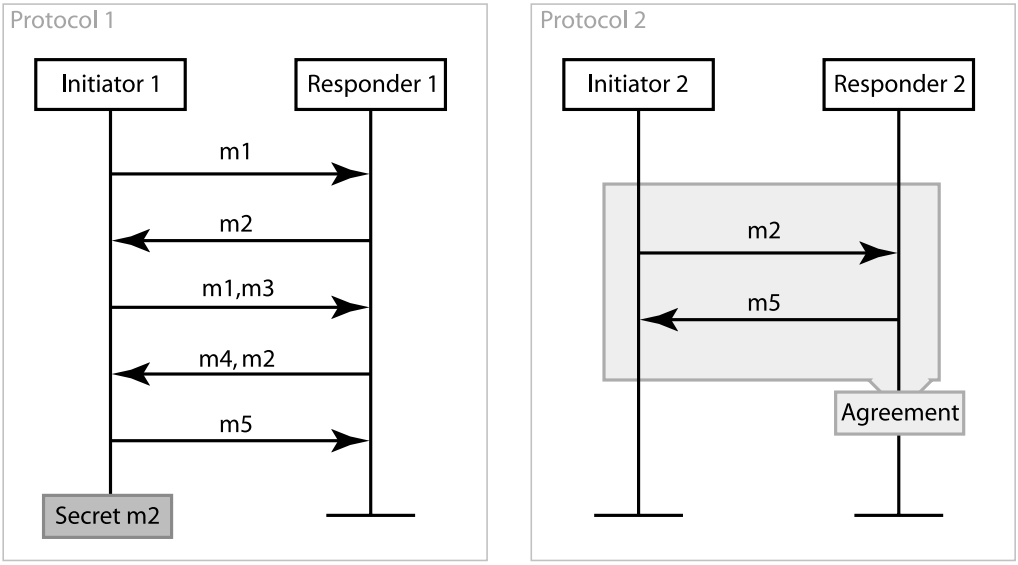

Fig. 1. A composition of two protocols

Consider the composition of the two protocols in figure 4.1. The message $m2$ is claimed to be secret in protocol 1. However, it also occurs in protocol 2. A goal such as secrecy is global to composition of protocols, and possibly has implications for every message sent in the system. Thus, we would expect a secrecy primitive not to consist of a sub-protocol, but as a global requirement, or rather as a transformation rule on all messages.

Other goals such as session authentication can be constructed as protocols, which can encapsulate other messages or protocols. As indicated by the grey area in in protocol 2 in the same figure, we might indicate exactly which parts of the protocol are to be authenticated. Thus, if we encapsulate the sending of some data within the session authentication protocol, the data itself will be authenticated as well. An implementation of such a primitive could involve setting up a session key, and encrypting the data (along with a sequence number) with this session key.

4.2 Analysis of primitives

Once appropriate primitives are chosen, we need to analyse their compositional properties. It is desirable to have primitives with compositional properties that are as simple as possible. There is a trade-off between the complexity of compositional properties and efficiency/size. For example, when no data is shared between primitives, compositionality is easier than with shared key infrastructures.

When designing primitives, we expect dependencies to arise. For example, in the Dolev-Yao model, authentication requires the secrecy of some data items. Thus, a dependency is established between such the authentication concept and the secrecy concept, with respect to a specific intruder model. Investigating these dependencies will help in proving properties of the primitives and also help to make to descriptions more concise.

5 Related work

There exist other attempts at handling security protocol composition. We briefly discuss some attempts.

Although the majority of security protocol formalisms do not cope with protocol composition, there has been much work on improving the scalability of the current methods to larger protocols. We feel that this helps to pave the way for the verification of composed protocols. We mention two such methods. First, many advances have been made in the Strand Spaces model from [16], resulting in the efficient Athena tool in [13]. Second, a tool that is already able to deal with parallel protocols is the Scyther tool we developed. This tool is based on the operational semantics of security protocols from [6] and uses partial order reduction techniques as explained in [5].

A recent development is the concept of Universal Composability as proposed by Canetti in [3]. This approach has its roots in secure multi-party computation, and is tailored towards constructing (sub)protocols that achieve their goals, irrespective of the environment they are placed in. This leads to some stringent requirements on the protocols, but also on the environment. The resulting protocols differ in two important ways from the type of primitives we propose. First, they allow for a specific compositional property (universal composability) whereas we allow for more types of primitives, which possibly share information or encapsulate other protocols. Second, within the Universal Composability framework, there is a fixed intruder model, whereas we propose to formulate the concepts, goals and abstract primitives regardless of the intruder model.

Another approach has been to modify the Strand Spaces model, making it

suitable for handling protocol composition, in [15]. This Mixed Strand Spaces model has been used to derive theoretical results. Initial results showed how attacks on protocol composition were possible. Later results include circumstances under which protocols can be safely combined in [9], where suggestions for fixing the protocols are also given. This approach helps towards extending current methods, but it lacks a more appropriate level of abstraction.

6 Closing comments

The last few decades of security protocol research have resulted in impressive progress, in both the formal verification of fairly small protocols as well as tool support. The next step is to tackle larger protocols, composed of several sub-protocols.

To do any kind of formal analysis, some assumptions on the model are needed, which will always imply some form of closed world assumption. However, given the advances in the field, it should be possible to reason about a larger closed world.

Here we have presented our ideas on the research that is needed, for formal methods to be applicable to the composition of security protocols. We have also sketched initial ideas for another abstraction layer. Furthermore, we have discussed specific research topics that can serve as a starting point for future research.

Work that we have already done ourselves includes the development of an Operational Semantics of Security Protocols that supports protocol composition. Based on these semantics we have developed a verification tool, Scyther, that can find attacks on composed protocols.

However, much work still needs to be done. We summarise the main open research issues:

(i) Adapt existing semantics and tools to support protocol composition.

(ii) Determination of minimal upper bound on runs involved in an attack, given a protocol.

(iii) Investigate compositionality of proofs.

(iv) Extend specifications of security protocols, to include exact usage conditions, under which they are correct when e.g. composed with other protocols.

(v) Develop another abstraction layer for security protocol composition, and abstract away from protocol details.

(vi) Develop concepts such as security goals and primitives for this abstraction level.

(vii) Determine relations between these concepts, and the intruder model.

It is our intention to work on these research questions, and we would like to invite others to investigate them as well.

References

[1] Benaloh, J., B. Lampson, D. Simon, T. Spies and B. Yee, *The private communication technology protocol* (1995), draft-benaloh-pct-00.txt.

[2] Burch, J.R., E.M. Clarke, K.L. McMillan, D.L. Dill and L.J. Hwang, *Symbolic Model Checking: 10^{20} States and Beyond*, in: *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science* (1990), pp. 1–33, citeseer.lcs.mit.edu/burch90symbolic.html.

[3] Canetti, R., *Universally composable security: A new paradigm for cryptographic protocols*, Cryptology ePrint Archive, Report 2000/067 (2000), eprint.iacr.org/.

[4] Cremers, C., *Scyther documentation*, www.win.tue.nl/~ccremers/scyther.

[5] Cremers, C. and S. Mauw, *Checking secrecy by means of partial order reduction*, in: D. Amyot and A. Williams, editors, *SAM 2004: Security Analysis and Modelling*, 4th Workshop on SDL and MSC (2004), pp. 177–194.
URL http://www.win.tue.nl/~ecss/downloads/cm_secrecyreduction.pdf

[6] Cremers, C. and S. Mauw, *Operational semantics of security protocols*, in: *Dagstuhl 03371 post-seminar proceedings: Scenarios: Models, Algorithms and Tools*, LNCS, 2004, to be published.

[7] Cremers, C., S. Mauw and E. de Vink, *Defining authentication in a trace model*, in: T. Dimitrakos and F. Martinelli, editors, *FAST 2003*, Proceedings of the first international Workshop on Formal Aspects in Security and Trust (2003), pp. 131–145, www.win.tue.nl/~ecss/downloads/cmv_defining_authentication.ps.

[8] Dolev, D. and A. Yao, *On the security of public key protocols*, IEEE Transactions on Information Theory **IT-29** (1983), pp. 198–208.

[9] Guttman, J. D. and F. J. Thayer, *Protocol independence through disjoint encryption*, in: *PCSFW: Proceedings of The 13th Computer Security Foundations Workshop* (2000), citeseer.ist.psu.edu/guttman00protocol.html.

[10] Lowe, G., *Breaking and fixing the Needham-Schroeder public-key protocol using FDR*, Proceedings of TACAS **1055** (1996), pp. 147–166,
URL web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Papers/NSFDR.ps.

[11] Lowe, G., *Casper: A compiler for the analysis of security protocols*, in: *Proc. 10th Computer Security Foundations Workshop* (1997), pp. 18–30.

[12] Lowe, G., *A hierarchy of authentication specifications*, in: *Proc. 10th Computer Security Foundations Workshop* (1997), pp. 31–44.

[13] Song, D., *Athena: a new efficient automatic checker for security protocol analysis*, in: *Proceedings of the 1999 IEEE Computer Security Foundations Workshop* (1999), p. 192.

[14] *Security protocols open repository*, www.lsv.ens-cachan.fr/spore.

[15] Thayer, F. J., J. C. Herzog and J. D. Guttman, *Mixed strand spaces*, in: *Proceedings of the 1999 IEEE Computer Security Foundations Workshop* (1999), p. 72.

[16] Thayer, F. J., J. Herzog and J. Guttman, *Strand spaces: Proving security protocols correct*, Journal of Computer Security **7** (1999), pp. 191–230.