Composable Formal Security Analysis: Juggling Soundness, Simplicity and Efficiency

Ran Canetti*

IBM Research

Abstract. A security property of a protocol is *composable* if it remains intact even when the protocol runs alongside other protocols in the same system. We describe a method for asserting composable security properties, and demonstrate its usefulness. In particular, we show how this method can be used to provide security analysis that is formal, relatively simple, and still does not make unjustified abstractions of the underlying cryptographic algorithms in use. It can also greatly enhance the feasibility of *automated* security analysis of systems of realistic size.

1 Introduction

Security analysis of protocols is a slippery business. On the one hand, we want to capture all "feasible attacks". On the other hand, we want to allow those protocols that do not succumb to attacks. Indeed, time and again attacks are found against protocols that were thoroughly analyzed and sometime even deployed and standardized (see e.g. [Ble98,Low96]). The situation is particularly tricky when the analyzed protocol uses "cryptographic primitives", namely algorithms that guarantee certain behaviors only when the adversarial components of system are computationally bounded.

A crucial first step in any rigorous security analysis is to devise an appropriate mathematical model for representing protocols and formulating the desired security properties. Indeed, the analysis can only be meaningful to the degree that the devised model and the formulated security requirements are meaningful.

Many models for analyzing security of protocols have been proposed over the past few decades, each with its own advantages and drawbacks. Roughly, there are two main analytical approaches, which differ in the way the cryptographic primitives used by the protocol and their security properties are modeled. In symbolic models, devised mainly within the formal analysis community, cryptographic primitives are treated as abstract, or symbolic operations with rigid interfaces that restrict the way in which the primitives can be used - and, more importantly, the ways in which the primitives can be *mis*used by adversarial components. In a way, this models the cryptographic primitives in use as "ideal

* IBM T.J. Watson Research Center. Email: canetti@csail.mit.edu. Supported by NSF grant CFF-0635297 and US-Israel Binational Science Foundation Grant 2006317.

boxes" that provide "absolute security". Quintessential examples of such models include the Dolev-Yao model [DY83], the BAN logic [BAN89], Spi-calculus [AG97] and their many derivatives.

In contrast, computational models (such as those of [GM84,GMR89,BR93] and many others) explicitly treat cryptographic constructs as algorithms, and consider adversaries that have full access to the actual input and output strings of these algorithms. In that respect, these models directly reflect the actual capabilities of adversaries in realistic systems. Here, meaningful formalizations of security requirements have to be probabilistic. Furthermore, they have to incorporate computational bounds on the adversarial entities involved. In addition, given the current state of the art in complexity theory, such analysis has to rely on computational hardness assumptions.

These two analytical approaches provide a clear tradeoff: The symbolic approach is much simpler and easier to work with than the computational approach. Also, it is conceptually attractive since it allows for clear separation of the analysis at the "protocol level" from the analysis of the underlying primitives. However, at least a priori, the computational approach is the only one that is sound; that is, it is the only approach that can actually provide security guarantees for protocols in realistic settings.

A recent research program, initiated in [AR02] and followed in many works since, is aimed at combining these two analytical approaches in a single model that provides the best of both: Soundness together with the ability to argue about protocols in a symbolic and mechanical way. A number of approaches have been proposed to carry out this combination. This paper reviews one such approach, that builds on security models that provide a general *security-preserving composition* guarantee. Specifically, the approach uses the universal composition theorem [PW00,BPW04,Can01], which guarantees that a protocol that uses an abstractly specified primitive can be securely "composed" with a protocol that realizes this specification, "without bad side effects". Here the symbolic model would correspond to the protocol that uses the abstract primitive, and the soundness would follow from the security preserving composition theorem. See more details within.[1]

While the initial thrust of the above work is to argue the soundness of the symbolic approach, there is an additional aspect here that we wish to highlight. (Indeed, this aspect seems to have been overlooked by most works in that area.) The symbolic approach, being dramatically simpler than the computational one, lends naturally to mechanization and automation of the analysis (see e.g. [Mea96,MMS97,Bla03]). Still, traditional automated symbolic analysis is feasible only for relatively small systems: the complexity of analysis is typically exponential in the number of variables, parties, and protocol instances in the analyzed system (see e.g. [MS01]). In fact, when the protocol descrip-

[1] Our notion of "secure composition" differs from other notions of compositionality, such as the one in, say, [DMP01], which is a more fine-grained approach for synthesizing protocols from elementary instructions, and does not carry composition theorems akin to the ones here.

tion and number of instances is taken to be part of the input, the question whether a symbolic protocol satisfies a certain property is NP-hard. When the number of protocol instances is unbounded the question becomes undecidable [EG83,DLMS99].

Composable security can help overcome this complexity barrier in certain systems of interest. Indeed, when asserting composable security properties, it suffices to apply the symbolic analysis to small, single-instance systems. Security of large composite systems would then follow from the composition theorem.

Organization. This paper is organized as follows. Section 2 briefly reviews symbolic protocol analysis. Section 3 briefly reviews the universally composable (UC) security framework. Section 4 reviews ways in which UC security has been used to assert the soundness of symbolic analysis and to enable its efficient automation. Section 5 concludes with some directions for further research.

Throughout, we do not attempt to give a broad survey of all relevant works. Rather, the intention is to present the main ideas, concerns and challenges, as seen by the author, in a way that is accessible to the non-expert. We apologize for any misrepresentations and omissions.

2 Symbolic analysis in a nutshell

There are a number of approaches for formulating models for protocol analysis where the cryptographic primitives are represented in an "idealized", or abstract way. Examples include the Dolev-Yao model [DY83], which essentially amounts to formulating a protocol-dependent abstract algebra where security properties translate to representability questions in the algebra (see, e.g. [Pau98,FHG98]); the BAN logic [BAN89] where security properties are translated to assertions in a protocol-dependent logic; or the spi-calculus [AG97] where security properties are translated to observational equivalence assertions in an extension of the π-calculus [Mil89]. Here we briefly sketch one of these approaches, namely the Dolev-Yao model, which is relatively simple and self contained. (On the down side, this model tends to be specific for a given class of tasks and protocols, and has to be reformulated whenever the task or class of protocols changes.)

The Dolev-Yao model has several components. First, the model defines a symbolic algebra. The atomic elements of the algebra represent primitive structures such as party identifiers, public and secret keys for the cryptographic algorithms in use, and random challenges (nonces).

Operations in the algebra represent the allowed usage of the cryptographic primitives — both by the legitimate protocol parties and by adversarial entities. For instance, in the case of public key encryption the symbolic encryption operation Enc takes a public key symbol ek and arbitrary symbol m (say, an identifier or a nonce) to return a compound "ciphertext" symbol $Enc_{ek}(m)$. The symbolic decryption operation Dec takes a private key symbol dk and a symbol of the from $Enc_{ek}(m)$ where ek and dk are paired, and returns the symbol m.

Inputs, outputs, and protocol messages are represented as compound elements in the algebra. That is, each message (compound element) represents a "parse tree", or the sequence of operations needed to obtain the compound element from elementary ones. The algebra is free: it admits no equalities other than the identity. That is, each message has exactly one representation. In the above example, for instance, this means that there is no way to retrieve the symbol m (or gain any information on it) from $Enc_{ek}(m)$ without explicitly using the special symbol dk.

Symbolic protocols are defined via a function from the sequence of messages received so far to the next move, when a move consists of a message to be transmitted or alternatively some local output. All inputs, outputs, and messages are compound elements from the algebra.

The symbolic adversary is defined in two parts: its initial knowledge (a set of symbolic messages), and the adversary operations it can use to deduce new messages from known ones. (These known messages consist of the initial knowledge and the messages sent during the protocol execution.) The adversary operations are bound by the operations specified in the algebra. Typically, these operations are limited to the operations that represent the cryptographic primitives in use, plus simple operations such as concatenation and de-concatenation.

The closure of a message (or a set of messages) is the set of all messages that the adversary can potentially derive from the given message (or set). That is, the closure operation defines the messages which the adversary can create and transmit at any point.

A protocol execution in this model consists of a sequence of events where each event consists of the delivery of an adversarially generated message to some party, followed by the generation of new outgoing message, or a new local input, by that party.

The trace of an execution is the sequence of these events. The security properties of protocols are typically (but not always) predicates on sets of traces: A protocol satisfies such a security property if the predicate is satisfied by the set of that protocol's possible (or valid) traces.

As discussed in the introduction, this model has two substantial limitations: First, it does not provide any guarantees regarding the security of protocols that use concrete algorithms to implement the abstract cryptographic primitives postulated by the algebra. Second, mechanic verification of security properties of protocols is intractable in general. We'll see that both of these limitations can be overcome by taking a compositional approach to security analysis.

3 Universally Composable Security

We turn to a brief review of the universally composable (UC) security framework. (The first variant of the framework appears in [Can01]; some context and related work are briefly discussed below). The framework takes the cryptographic approach to protocol analysis; namely, the adversarial entities are given unre-

stricted access to the actual bits of the communication between parties. Also, adversaries are taken to be computationally bounded and the security properties are stated in probabilistic and asymptotic terms.

In this setting, the framework provides a general way for specifying the security requirements of cryptographic tasks, and asserting whether a given protocol realizes the specification. A salient property of this framework is that it provides strong composability guarantees: A protocol that meets a specification in isolation continues to meet the specification regardless of the activity in the rest of the network. We give here a very high level sketch of the framework, as well as some motivation. See [Can01,Can06] for a more thorough treatment.

The trusted party paradigm. The underlying definitional idea (which originates in [GMW87], albeit very informally) proceeds as follows. To determine whether a given protocol is secure for some cryptographic task, first envision an ideal process for carrying out the task in a secure way. In the ideal process all parties secretly hand their inputs to an external trusted party who locally computes the outputs according to the specification, and secretly hands each party its prescribed outputs. This ideal process can be regarded as a "formal specification" of the security requirements of the task. (For instance, when the task is to compute a joint function f of the local inputs of the parties, the trusted party simply evaluates f on the inputs provided by the parties, and hands the outputs back to the parties. If the function is probabilistic then the trusted party also makes the necessary random choices.) The protocol is said to securely realize a task if running the protocol amounts to "emulating" the ideal process for the task, in the sense that any damage that can be caused by an adversary interacting with the protocol can also happen in the ideal process for the task.

An attractive property of this approach is its generality: It seems possible to capture the requirements of very different tasks by considering different sets of instructions for the external trusted party. Another attractive property is potential compositionality: It seems almost "built into the definitional approach" that if a protocol successfully mimics the behavior of some trusted party then any protocol that uses the protocol should continue to behave the same when the protocol is replaced by the trusted party.

Still, substantiating this approach in a way that maintains its intuitive appeal and materializes the potential generality and composability turns out to be non-trivial. Indeed, several general frameworks for representing cryptographic protocols and specifying the security requirements of tasks were developed over the years, e.g. [GL90,MR91,Bea91,Can00,HM00,DM00,PW00,Can01,PMS03,K06]. While all of these frameworks follow the above paradigm in one way or another, they differ greatly in their expressibility (i.e., the range of security concerns and tasks that can be captured), in the computational models addressed, and in many significant technical details. They also support different types of security-preserving composition theorems.

The basic formalism. Defining what it means for a protocol π to "securely realize" a certain task is done in three steps, as follows. First, we formulate a model for executing the protocol. This model consists of the parties running

π, plus two adversarial entities: the environment $\mathcal{Z}$, which generates the inputs for the parties and reads their outputs, and the adversary $\mathcal{A}$, which reads the outgoing messages generated by the parties and delivers incoming messages to the parties. The adversary and the environment can interact freely during the protocol execution.

The adversary represents attacks against a single instance of the analyzed protocol. The environment represents "everything else that happens in the system," including both the the immediate users of the protocol, and other parties and protocols. Letting $\mathcal{A}$ and $\mathcal{Z}$ interact freely during the computation represents the continual information flow between an execution of a protocol and the rest of the system. Indeed, this provision turns out to be critical for the universal composition theorem to hold.

Next, we formulate the ideal process, in a straightforward way. Here the protocol participants simply pass their inputs to an additional, incorruptible *trusted party,* who locally computes the desired outputs and hands them back to the parties. The program run by the trusted party is called an ideal functionality and is intended to capture the security and correctness specifications of the task. For convenience, the ideal process with ideal functionality $\mathcal{F}$ is formulated as the process of running a special protocol $I_{\mathcal{F}}$ called the ideal protocol for $\mathcal{F}$. That is, in protocol $I_{\mathcal{F}}$ the parties simply pass all inputs to the trusted party, and output whatever information they obtain from the trusted party. Here the adversary does not interact directly with the parties; instead, it interacts with $\mathcal{F}$ in a way specified by $\mathcal{F}$. The communication between the adversary and the environment remains arbitrary.

Finally, we say that protocol π UC-emulates protocol ϕ if for *any* polytime adversary $\mathcal{A}$ there *exists* a polytime adversary $\mathcal{S}$ such that no polytime environment $\mathcal{Z}$ can tell with non-negligible probability whether it is interacting with an execution of π and adversary $\mathcal{A}$, or alternatively with protocol ϕ and adversary $\mathcal{S}$. We say that π UC-realizes an ideal functionality $\mathcal{F}$ if it UC-emulates the ideal protocol $I_{\mathcal{F}}$. Somewhat more formally, let $\{\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi}\} = \{\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi}(n,z)\}_{n\in N, z\in\{0,1\}^n}$ denote the probability ensemble describing the output of environment $\mathcal{Z}$ in an interaction with adversary $\mathcal{A}$ and protocol π with security parameter n and external input z for $\mathcal{Z}$. Then:

Definition 1. *Protocol π* UC-emulates *protocol ϕ if for any polytime adversary $\mathcal{A}$ there exists a polytime adversary $\mathcal{S}$ such that for any polytime environment $\mathcal{Z}$ we have* $\text{Prob}(\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi} = 1) - \text{Prob}(\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi} = 1)| < \nu(n)$*, where ν is a negligible function.*

π UC-realizes *an ideal functionality $\mathcal{F}$ if it UC-emulates the ideal protocol $I_{\mathcal{F}}$.*

Very informally, the goal of the above requirement is to guarantee that any information gathered by the adversary $\mathcal{A}$ when interacting with π, as well as any "damage" caused by $\mathcal{A}$, could have also been gathered or caused by an adversary $\mathcal{S}$ in the ideal process for $\mathcal{F}$. Now, since the ideal process is designed so that *no* $\mathcal{S}$ can gather information or cause damage more than what is explicitly permitted in the ideal process for $\mathcal{F}$, we can conclude that $\mathcal{A}$ too, when interacting with

π, cannot gather information or cause damage more than what is explicitly permitted by $\mathcal{F}$. In particular, the I/O behavior of the good parties in the protocol execution is essentially the same as that of the ideal functionality; similarly, the information that $\mathcal{Z}$ learns from $\mathcal{A}$ can be generated (or, "simulated") by $\mathcal{S}$, who is given only the information that it can learn legally from interacting with $\mathcal{F}$.

We remark that the notion of UC emulation can be viewed as a relaxation of the notion of *observational equivalence* of processes (see, e.g., [Mil89]); indeed, observational equivalence essentially fixes the entire system outside the protocol instances, whereas emulation allows the analyst to choose an appropriate simulator that will make the two systems look observationally equivalent. In a way, this relaxation allows the analyst to specify which properties of the analyzed protocol are "salient" and which are "unimportant", and thereby allow for many proofs of security of cryptographic protocols to go through.

Universal composition. The following universal composition theorem holds in this framework. Let π be a protocol that UC-emulates protocol ϕ, and let ρ be a protocol that has access to (multiple instances of) ϕ. Let $\rho^{\pi/\phi}$ be the "composed protocol" which is identical to ρ except that inputs to ϕ are replaced by inputs to π, and outputs from π are treated as outputs from ϕ. Then, protocol $\rho^{\pi/\phi}$ behaves in an indistinguishable way from the original ρ:

Theorem 1. *Let ρ, π, ϕ be protocols such that π UC-emulates ϕ. Then $\rho^{\pi/\phi}$ UC-emulates ρ.*

4 Composable Formal Security Analysis

Providing soundness. The idea underlying the use of security-preserving protocol composition for asserting soundness of formal analysis is simple: Intuitively, the formal (or, symbolic) model appears to naturally correspond to a model where protocols have access to a "trusted party") that embodies the abstract properties of the cryptographic primitives in use, just as in the definition of UC realization. Thus, the universal composition theorem should imply that any security property enjoyed by the symbolic protocol continues to be enjoyed by the protocol even when the symbolic cryptographic primitive is replaced by a concrete protocol that realizes the corresponding ideal functionality. This idea was mentioned already in [PW00] with respect to their formalism (which bears some significant similarities with the UC framework) and also in [Can01].

Substantiating this idea involves a number of steps. Specifically, one has to carry out the following:

1. Formulate ideal functionalities, within the UC framework, that capture in an abstract way the functionality and security properties of the cryptographic primitives in use.
2. Devise concrete protocols that UC-realize the formulated functionalities.
3. Formulate a class (or, rather, a "programming language") of concrete protocols that make use of (i.e., subroutine calls to) the formulated ideal func-

tionalities. (We call these protocol hybrid protocols, since they are a hybrid of a concrete protocol with an abstract ideal functionality.)
4. Formulate a symbolic model that models the cryptographic primitives in use in an abstract way, akin to the formulated ideal functionalities.
5. Formulate a security property (goal) for the concrete protocols.
6. Formulate a translation of this property in the symbolic model, and a method for asserting this property in the symbolic model.
7. Demonstrate that if a symbolic protocol satisfies the symbolic property (within the symbolic model) then the corresponding hybrid protocol, within the devised language, satisfies the corresponding concrete property.

Now, we can translate hybrid protocols to fully concrete ones by replacing the ideal functionalities with the protocols that UC-realize them, and use the universal composition theorem to deduce that the fully concrete protocols enjoy the same security properties enjoyed by the hybrid protocols.

A substantiation of these ideas, along the lines of the above sketch, is given in [BPW03]. That work concentrates on protocols where the cryptographic primitives in use are public-key encryption, digital signatures, and secure communication channels. They also provide symbolic constructs that correspond to the use of random challenges, or nonces. (Indeed, these primitives are the ones addressed by traditional symbolic models.)

Specifically, an ideal functionality is formulated, that provides the interface expected from the above primitives, along with absolute security properties. For instance, to model public-key encryption, the [BPW03] ideal functionality provides an encryption interface, that takes a public key symbol and a message and returns an abstract handle, and a decryption interface that takes a handle and a decryption key symbol, and returns the message associated with the handle and the corresponding encryption key - in case these are defined. (Else the decryption interface returns an error symbol.) Digital signatures and secure channels are modeled via handles in a similar way.

Next, [BPW03] show that their ideal functionality can be realized using known cryptographic protocols. Specifically, any combination of an encryption scheme that's semantically secure against chosen ciphertext attacks [RS91,DDN00] with a signature scheme that's existentially unforgeable against chosen message attacks, along with appropriate symmetric encryption and authentication schemes (for obtaining secure communication channels), suffice.

This work opens the door for abstract security analysis of protocols that use the above primitives. All there is to do is to write the protocol in a way that uses the [BPW03] ideal functionality for all its cryptographic operations. Now, the protocol becomes considerably simpler; in fact, in many cases it becomes deterministic, akin to the symbolic ("Dolev-Yao") model. Security of the corresponding concrete protocol follows from the universal composition theorem, as discussed above.

We note that the [BPW03] modeling does not formulate a dedicated abstract model along the lines of the original Dolev-Yao analysis. Instead, even the abstract protocols are defined and analyzed in the same cryptographic model

in which the full-fledged cryptographic protocols are. This forces the analyst to either analyze the abstract protocol in a relatively complex and model, or alternatively simplify the model at the price of reduced generality in terms of expressing realistic concerns and situations.

Still, this approach has proven to be very useful, serving as a basis for analyzing a number of protocols, e.g. [Bac04,BP04,BD05,Bac06,BP06,BCJ+06]. The security properties asserted in these works are mainly mutual authentication and generation of a common secret key ("key exchange"), as well as other properties such as transactional integrity in payment systems. Also, this work has been the basis for semi-automated security analysis of protocols, using the Isabelle theorem prover [SBB+06,Pau88].

Feasible mechanization and automation. So far, we have seen how to perform symbolic (abstract) security analysis that provides security guarantees even for fully concrete protocols. However, in spite of its apparent simplicity, traditional symbolic analysis still has a serious shortcoming: As argued in the introduction, performing such analysis in a fully mechanical (or, automated) way is intractable, even for systems of moderate size. Consequently, we can feasibly analyze in a fully mechanical way only systems of relatively small size. In particular, we cannot directly analyze systems which consist of unboundedly many concurrent protocol instances, even when all these instances are instances of the same protocol.

Also here, composable security offers a natural solution: When coming to analyze security of a complex system, first de-compose the system to relatively small components; then, use symbolic analysis to mechanically analyze the security of each component; finally, use the composition theorem to re-compose the components and deduce security properties of the whole system. In particular, when coming to analyze a system which consists of an unbounded number of sessions of the same protocol, it suffices to analyze a single session of this protocol, in isolation.

Two main issues need to be addressed in order to make good of this approach, when carrying out the steps described above: First, in order to be able to perform the symbolic analysis separately in each component, independently of all other components, the ideal functionality in Step 1 above needs to be "de-composable" into multiple independent, simpler ideal functionalities, where each such simpler functionality is used only within a single component.

Second, in order to be able to deduce a security property of the re-composed system from the security of the individual components, the security properties asserted by the symbolic analysis (see Step 6 above) needs to be phrased as *composable* security properties. (In the UC framework, this means that the symbolic security properties need to be translatable to assertions of UC-realizing some ideal functionalities.)

A first attempt for coming up with a formalism that addresses the above two concerns would be to try to use the [BPW03] formalism described above. However, it turns out that this formalism does not address the first concern. (Indeed, here it seems essential that all instances of all cryptographic algorithms

will reside within a single ideal functionality. See more discussion in [Can04]). Furthermore, the security properties asserted within this formalism (see above literature) are not composable; thus the second concern is not addressed either.

We are thus motivated to look for alternative ways to substantiate the composable approach to symbolic analysis, that will allow us to materialize the prospective efficiency gains. Such an alternative approach is given in [CH04]. That work concentrates on protocols that use a single cryptographic primitive, namely public-key encryption. As in [BPW03], an ideal functionality is presented that captures the behavior of ideal encryption. Here, however, the formalism is such that multiple instances of the ideal functionality can co-exist in the same system where each instance represents encryption via a different set of keys. (On a technical level, this change requires, among other things, abandoning the convenient abstraction of "handles;" instead, the ideal functionality returns "dummy ciphertexts", which are strings generated by an adversarial computational entity without knowledge of the plaintext.) Still, it is shown in [CH04] that any public-key encryption scheme that's semantically secure against chosen ciphertext attacks can be used to UC-realize the devised ideal functionality. This addresses the first concern mentioned above.

The security properties asserted in [CH04] are the traditional ones: Mutual Authentication and Key Exchange. However, in order to address the second concern, these properties are formulated as composable security properties. Specifically, in [CH04] a special-purpose symbolic algebra is devised for representing the class of protocols considered. Next, symbolic Mutual Authentication and Key Exchange properties are formulated. It it then shown that a symbolic protocol satisfies the symbolic Mutual Authentication (resp., symbolic Key Exchange) property *if and only if* the corresponding concrete protocol UC-realizes an ideal Mutual Authentication (resp., Key Exchange) functionality.

To demonstrate the validity of their approach, [CH04] encode the devised symbolic properties in the language of the ProVerif verification tool [Bla03], and use it to automatically assert security of a systems consisting of an unbounded number of concurrent instances of some variants of the Needham-Schroeder-Lowe protocol. The analysis takes less than a second on a standard commodity laptop. We remind the reader that directly analyzing such a system using traditional means is undecidable.

We remark that [CH04] is strongly influenced by [MW04]. In fact, for the case of mutual authentication [CH04] follows the approach of [MW04] quite closely. However, [MW04] is not formulated within a composable framework and thus it cannot provide the efficiency gains provided by [CH04].

5 Future research

We are at the early stages of capitalizing on the potential of composable notions of security in enabling sound automated analysis of complex systems. Directions for further research include:

1. Widen the range of cryptographic primitives that can be modeled in an abstract, symbolic, and composable way. In the same vein, widen the range of security properties and tasks that can be asserted symbolically.
2. Construct new tools (or, improve existing ones) to allow for efficient automated security analysis, capitalizing on the composable approach to analysis, with the end goal being to perform fully automated security analysis of real-life systems. An interesting challenge here is to mechanize the process of de-composing a system to small components.
3. In a slightly different vein, it might be interesting to formulate and assert the composability of security properties directly in a symbolic model, without having to rely on the composability properties of the underlying computational framework.

Acknowledgements. I thank the program committee of ICALP 2008 for inviting me to talk at the conference and for soliciting this paper. Special thanks is also due to Oded Goldreich for his invaluable conceptual advice and direction.

References

[AG97] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *4th ACM Conference on Computer and Communications Security, pp.36-47. See also http://www.research.digital.com/SRC/ abadi.*, 1997.

[AR02] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptology 15(2): 103-127*, 2002.

[Bac04] M. Backes. A cryptographically sound Dolev-Yao style security proof of the Otway-Rees protocol. *ESORICS: 89-108*, 2004.

[Bac06] M. Backes. Real-or-random key secrecy of the Otway-Rees protocol via a symbolic security proof. *Electr. Notes Theor. Comput. Sci. 155: 111-145*, 2006.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic for authentication. *DEC Systems Research Center Technical Report 39, February 1990. Earlier versions in the Second Conference on Theoretical Aspects of Reasoning about Knowledge, 1988, and the Twelfth ACM Symposium on Operating Systems Principles,*, 1989.

[BCJ+06] M. Backes, I. Cervesato, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Cryptographically sound security proofs for basic and public-key Kerberos. *ESORICS: 362-383*, 2006.

[BD05] M. Backes and M. Dürmuth. A cryptographically sound Dolev-Yao style security proof of an electronic payment system. *CSFW: 78-93*, 2005.

[Bea91] D. Beaver. Foundations of secure interactive computing. *CRYPTO '91, LNCS 576,*, 1991.

[Bla03] B. Blanchet. Automatic proof of strong secrecy for security protocols. *IEEE Security and Privacy Conference, pages 86–102.*, 2003.

[Ble98] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. *CRYPTO*, pages 1–12, 1998.

[BP04] M. Backes and Birgit Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. *IEEE Journal on Selected Areas in Communications 22(10): 2075-2086*, 2004.

[BP06] M. Backes and B. Pfitzmann. On the cryptographic key secrecy of the strengthened Yahalom protocol. *SEC: 233-245*, 2006.

[BPW03] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. *10th ACM CCS. Extended version at http://eprint.iacr.org/2003/015/*, 2003.

[BPW04] M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. *1st Theory of Cryptography Conference (TCC), LNCS 2951 pp. 336–354*, 2004.

[BR93] M. Bellare and P. Rogaway. Entity authentication and key distribution. *CRYPTO '93*, pages 232–249, 1993. Full version of paper available at http://www-cse.ucsd.edu/users/mihir/.

[Can00] R. Canetti. Security and composition of multi-party cryptographic protocols. *J. Cryptology, Vol. 13, No. 1,*, 2000.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. *FOCS*, pages 136–145, 2001. Long version at IACR Eprint Archive entry 2000/067.

[Can04] R. Canetti. Universally composable signature, certification, and authentication. *CSFW. Long version at eprint.iacr.org/2003/239.*, 2004.

[Can06] Ran Canetti. Security and composition of cryptographic protocols: A tutorial. *SIGACT News, Vol. 37, Nos. 3 & 4. Available also at the Cryptology ePrint Archive, Report 2006/465*, 2006.

[CH04] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols (the case of encryption-based mutual authentication and key-exchange). *3rd TCC, 2006. Full version at Cryptology ePrint Archive, Report 2004/334*, 2004.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DLMS99] N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. *Workshop on Formal Methods and Security Protocols (FMSP)*, 1999.

[DM00] Y. Dodis and S. Micali. Secure computation. *CRYPTO '00*, 2000.

[DMP01] N.A. Durgin, J.C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. *SCFW*, 2001.

[DY83] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory, 2(29)*, 1983.

[EG83] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. *24th FOCS*, pages 34–39, 1983.

[FHG98] F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? *IEEE Symposium on Security and Privacy,*, 1998.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. *CRYPTO '90*, pages 77–93, 1990.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS, Vol. 28, No 2, pp. 270-299.*, 1984.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Comput., Vol. 18, No. 1, pp. 186-208*, 1989.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *19th Symposium on Theory of Computing (STOC), pp. 218-229*, 1987.

[HM00] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. *J. Cryptology, Vol 13, No. 1, pp. 31-60 4*, 2000.

[K06] R. Küsters. Simulation based security with inexhaustible interactive Turing machines. *19th CSFW*, 2006.

[Low96] G. Lowe. Breaking and fixing the Needham-Schröder public-key protocol using CSP and FDR. *2nd International Workshop on Tools and Algorithms for the construction and analysis of systems.*, 1996.

[Mea96] C. Meadows. The NRL protocol analyzer: An overview. *J. Log. Program.*, 26(2):113–131, 1996.

[Mil89] R. Milner. Communication and concurrency. *Prentice Hall*, 1989.

[MMS97] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Proceedings, 1997 IEEE Symposium on Security and Privacy*, pages 141–153, 1997.

[MR91] S. Micali and P. Rogaway. Secure computation (abstract). *CRYPTO '91*, pages 392–404, 1991.

[MS01] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. *ACM Conference on Computer and Communications Security (CCS)*, 2001.

[MW04] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. *1st TCC*, pages 133–151, 2004.

[Pau88] L. C. Paulson. Isabelle: the next seven hundred theorem provers (system abstract). *9th International Conf. on Automated Deduction, LNCS 310, pp. 772-773. More details at http://www.cl.cam.ac.uk/Research/HVG/Isabelle/*, 1988.

[Pau98] L C Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[PMS03] J. C. Mitchell P. Mateus and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. *14th CONCUR, pp. 323-345*, 2003.

[PW00] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. *7th ACM Conf. on Computer and Communication Security (CCS), pp. 245-254.*, 2000.

[RS91] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *CRYPTO '91*, 1991.

[SBB+06] C. Sprenger, M. Backes, D. A. Basin, B. Pfitzmann, and M. Waidner. Cryptographically sound theorem proving. *CSFW: 153-166*, 2006.