Secure Pseudonymous Channels

Sebastian Mödersheim[1] Luca Viganò[2]

[1] IBM Zurich Research Laboratory, Switzerland, smo@zurich.ibm.com
[2] Dep. of Computer Science, University of Verona, Italy, luca.vigano@univr.it

Abstract. Channels are an abstraction of the many concrete techniques to enforce particular properties of message transmissions such as encryption. We consider here three basic kinds of channels—authentic, confidential, and secure—where agents may be identified by pseudonyms rather than by their real names. We define the meaning of channels *as assumptions*, i.e. when a protocol relies on channels with particular properties for the transmission of some of its messages. We also define the meaning of channels as *goals*, i.e. when a protocol aims at establishing a particular kind of channel. This gives rise to an interesting question: given that we have verified that a protocol P_2 provides its goals under the assumption of a particular kind of channel, can we then replace the assumed channel with an arbitrary protocol P_1 that provides such a channel? In general, the answer is negative, while we prove that under certain restrictions such a compositionality result is possible.

1 Introduction

Context In recent years, a number of works have appeared that provide formal definitions of the notion of channel and how different kinds of channels can be employed in security protocols and web services as a means of securing the communication. These works range from the definition of a calculus for reasoning about what channels can be created from existing ones [23] to the investigation of a lattice of different channel types [15]. In this paper, we consider three basic kinds of channels: authentic, confidential, and secure. We use an intuitive notation from [23], where a secure end-point of a channel is marked by a bullet with the following informal meaning (defined precisely below):

- $A \bullet\!\!\rightarrow B : M$ represents an *authentic channel* from A to B. This means that B can rely on that fact that A has sent the message M and meant it for B.
- $A \rightarrow\!\!\bullet\, B : M$ represents a *confidential channel*. This means that A can rely on that fact that only B can receive the message M.
- $A \bullet\!\!\rightarrow\!\!\bullet\, B : M$ represents a *secure channel*, i.e. a channel that is both authentic and confidential.

While [23] uses the bullet notation to reason about the existence of channels, we use it to specify message transmission in security protocols and web services in two ways. First, we may use channels *as assumptions*, i.e. when a protocol relies on channels with particular properties for the transmission of some of its messages. Second, the protocol may have the *goal* of establishing a particular kind of channel.

Contributions First, for channels as assumptions, we define two models: the *Ideal Channel Model ICM* describes the ideal functionality of a channel, and the *Cryptographic Channel Model CCM* describes the implementation of channels by cryptographic means. We relate these two models by showing that attacks in either model can be simulated in the other. On the theoretical side, relating ideal functionality and cryptographic implementation gives us insight in the meaning of channels as assumptions. On the practical side, it allows us to use the models interchangeably in analysis tools, which may have different preferences.

Second, we formally define the meaning of channels as goals. Specifying the use of channels both as assumptions and goals gives rise to an interesting question: given that we have verified that a protocol P_2 provides its goals under the assumption of a particular kind of channel, can we then replace the assumed channel with an arbitrary protocol P_1 that provides such a channel? In general, the answer is negative, while we prove that under certain restrictions such a compositionality result is possible.

On the theoretical side, this proof has revealed several subtle properties of channels that had not been recognized before, so we contribute to a clearer picture of channels and protocol goals. The most relevant issue is the following one. We discovered that the standard authentication goals that are widely used in formal protocol verification are too weak for our compositionality result, as we illustrate with a simple example protocol. We propose a strictly stronger authentication goal that, to our knowledge, has never been considered before and that is sufficient for compositionality.

On the practical side, such a compositionality result is vital for the verification of larger systems. For example, when using an application protocol on top of a protocol for establishing a secure channel such as TLS, one may try to verify this as one large protocol, but this has several drawbacks in terms of complexity and reuseability. With our approach, one can instead verify each of the two protocols in isolation and reuse the verification results of either protocol when employing them in a different composition, i.e. when using the channel protocol for a different application, and when running the application protocol over a different channel protocol.

Third, we formulate all the above channel models and theorems so that an agent may be identified not by its real name but by some pseudonym, which is usually related to an unauthenticated public-key; see, e.g., [7,14,18,20]. In the case of authentic channels, this concept has often been referred to as *sender invariance*: the receiver can be sure that several messages come from the same source, whose true identity is not known or not guaranteed. Analogously, one may consider *receiver invariance*.

The most common example of a pseudoynmous secure channel is TLS without client authentication: while the real name of the client is not authenticated (or not even mentioned), the established channel is secure but only relative to an unauthenticated agent. We show how to model this channel like a normal secure channel with a pseudonym instead of the agent's real name. Such a channel is sufficient for a number of applications, e.g. a login protocol where the unauthen-

ticated client sends a username and password to the server; this authentication turns the pseudonymous secure channel into a standard secure channel.

We proceed as follows. In § 2, we briefly describe the formal specification languages that we use. In § 3, we specify channels as assumptions and define and show equivalent the ICM and the CCM. In § 4, we specify channels as goals. In § 5, we consider compositional reasoning. In § 6, we discuss related work and draw conclusions. Proofs and further details can be found in [26].

2 The formal specification languages AnB• and IF

The definitions and results we present in this paper deal with the notion of secure pseudonymous channels in general, as employed in, or provided by, security protocols and web services. In this section, we give a brief overview of the *AVISPA Intermediate Format IF* that we use as a basis for our formalization. First, however, we introduce an extension of the language AnB [25], a formal language based on Alice and Bob notation for specifying security protocols, which we augment here with the bullet notation of [23] to easily specify secure channels as assumptions and goals; we call this extension AnB•. For lack of space, we omit several details of AnB• and IF, and of the translation from AnB• to IF, which can be found in [26].

AnB• Fig. 1 shows the AnB• specification of two example protocols that we use as running examples, where we omitted the declaration of types and initial knowledge for brevity. The protocol P (on the left) is the Diffie-Hellman key-exchange over authentic channels (as assumptions) plus a payload message symmetrically encrypted with the agreed key $\exp(\exp(g, X), Y)$, where we use $\{\!|\cdot|\!\}$. to denote symmetric encryption. Below the horizontal line, we have the goal that the payload message is transmitted securely. We may rephrase this protocol and (intended) goal as follows: Diffie-Hellman allows us to obtain a secure channel out of authentic channels. We have a similar setup in TLS, for instance, but we have selected this example for brevity.

Pseudonymous channels are like standard channels with the only exception that one of the secured endpoints is logically tied to a pseudonym instead of a real name. In general, we write $[A]_\psi$ to denote the identity of an agent A that is not identified by its real name A but by some pseudonym ψ, e.g. we write $[A]_\psi \bullet\!\!\rightarrow B : M$ for an authentic channel. We also allow that the specification of ψ is omitted, and write only $[A] \bullet\!\!\rightarrow B$, when the role uses only one pseudonym in the entire session (which is the case for most protocols). We use a similar notation for the other kinds of pseudonymous channels.

The protocol P' on the right of Fig. 1 is a variant of P where the message from A is on an insecure channel, thus A's half-key is not authenticated. We have here a weaker goal: a secure channel where A cannot be authenticated and is identified by a pseudonym. Again, we have a similar situation in the case of TLS without client authentication: we get a secure channel but the client is not authenticated. As follows from our compositionality result, such a pseudonymous

$$\frac{\begin{array}{rcl} A & \bullet\!\!\rightarrow & B : \exp(g, X) \\ B & \bullet\!\!\rightarrow & A : \exp(g, Y) \\ A & \rightarrow & B : \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \end{array}}{\begin{array}{rcl} A & \bullet\!\!\rightarrow\!\!\bullet & B : Payload \end{array}} \qquad \frac{\begin{array}{rcl} A & \rightarrow & B : \exp(g, X) \\ B & \bullet\!\!\rightarrow & A : \exp(g, Y) \\ A & \rightarrow & B : \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \end{array}}{\begin{array}{rcl} [A] & \bullet\!\!\rightarrow\!\!\bullet & B : Payload \end{array}}$$

Fig. 1. Example protocols in AnB• (excerpts): P (left) and the variant P'.

secure channel between an unauthenticated client and an authenticated server is sufficient to run, for instance, a password-based login protocol on it, such as

$$\frac{\begin{array}{rcl} [A] & \bullet\!\!\rightarrow\!\!\bullet & B : A, password(A) \\ [A] & \bullet\!\!\rightarrow\!\!\bullet & B : Payload' \end{array}}{\begin{array}{rcl} A & \bullet\!\!\rightarrow\!\!\bullet & B : Payload' \end{array}}$$

where $Payload'$ is now on a standard secure channel (assuming that the password of A is sufficient to authenticate her to the server B). We will continue our running examples below, giving concrete IF transition rules.

The Intermediate Format IF An *IF specification* $P = (I, R, G)$ consists of an *initial state* I, a *set* R *of rules* that induces a transition relation on states, and a *set* G *of attack rules* (i.e. *goals*) that specify which states count as attack states. A protocol is *safe* when no attack state is reachable from I using the transition relation. An IF state is a set of ground *facts*, separated by dots ("."), such as $\mathsf{iknows}(m)$, which expresses that the intruder knows m, or $\mathsf{state}_{\mathcal{A}}(A, m_1, \ldots, m_n)$, which characterizes the local state of an honest agent during the protocol execution by the messages $A, m_1, \ldots, m_n$. The constant $\mathcal{A}$ identifies the role of that agent, and, by convention, the first message A is the name of the agent. Note that state numbers are also messages and usually follow the agent name in state predicates (cf., e.g., (1) below). We will later introduce further kinds of facts.

The transition system defined by an IF specification consists of only ground states: the initial state is ground and transitions cannot introduce variables. We consider here IF transition rules of the form:

$$L \mid cond \;\text{=}[V]\Rightarrow\; R$$

where L and R are sets of facts, *cond* is a set of conditions of the form $\mathsf{not}(f)$ and $s \neq t$ for a fact f and terms s and t, and V is a list of variables that do not occur in L or *cond*; moreover, R may only contain variables that also occur in L or V. The semantics of this rule is defined by the state transitions it allows: we can get from a state S to a state S' with this rule iff there is a substitution σ of all variables of L and V such that $L\sigma \subseteq S$, $S' = (S \setminus L\sigma) \cup R\sigma$, and $V\sigma$ are fresh constants (that do not appear in S); moreover, for all substitutions τ

of the remaining variables that appear only in *cond*, the conditions are satisfied, i.e. $f\sigma\tau \notin S$ for each $\mathsf{not}(f) \in cond$, and $s\sigma\tau \not\approx t\sigma\tau$ for each $s \neq t \in cond$.

The transition rules of honest agents specify how agents reply to messages they receive. For instance, the second transition of A for our example protocol P of Fig. 1 looks as follows when using insecure channels:

$$\begin{aligned}\mathsf{state}_{\mathcal{A}}(A,1,B,g,X).\mathsf{iknows}(GY) =[Payload]\Rightarrow\\ \mathsf{iknows}(\{|Payload|\}_{\exp(GY,X)}).\mathsf{state}_{\mathcal{A}}(A,2,B,g,X,GY,Payload)\end{aligned} \tag{1}$$

By convention, all identifiers that start with an upper-case letter are variables, the others are functions. This rule describes the behavior of an agent A, playing role $\mathcal{A}$, in step 1 of the protocol execution: A has sent to agent B the first message of the protocol $\exp(g,X)$ and is waiting for the answer that corresponds to the $\exp(g,Y)$ step of the protocol. We adopt here an optimization for the case of insecure channels: we identify intruder and network for insecure channels (that are controlled by the intruder, see [24] for a soundness proof). Effectively, this means that the incoming message that A is waiting for is represented by an $\mathsf{iknows}(\cdot)$ fact (i.e. some value that the intruder chooses from his knowledge), and similarly the outgoing message is added directly to the intruder knowledge. Note that the left-hand side $\mathsf{iknows}(\cdot)$ fact does not need to be repeated on the right-hand side as we define $\mathsf{iknows}(\cdot)$ facts to be *persistent*. Since A cannot check that the value she receives is indeed of the form $\exp(g,Y)$ as the protocol says, she now accepts any value GY and will thus generate the full Diffie-Hellman key as $\exp(GY,X)$ and use it to symmetrically encrypt the *Payload*. Here, the *Payload* is modeled as a fresh nonce as a kind of place-holder; as we will see in § 5, there is actually a non-trivial verification problem attached to this.

We can describe the behavior of the intruder using similar rules; for this paper, we need the following deduction rules:

$$\begin{aligned}\mathsf{iknows}(M).\mathsf{iknows}(K) &\Rightarrow \mathsf{iknows}(\{M\}_K)\\ \mathsf{iknows}(\{M\}_K).\mathsf{iknows}(\mathsf{inv}(K)) &\Rightarrow \mathsf{iknows}(M)\\ \mathsf{iknows}(\{M\}_{\mathsf{inv}(K)}) &\Rightarrow \mathsf{iknows}(M)\\ \mathsf{iknows}(M).\mathsf{iknows}(K) &\Rightarrow \mathsf{iknows}(\{|M|\}_K)\\ \mathsf{iknows}(\{|M|\}_K).\mathsf{iknows}(K) &\Rightarrow \mathsf{iknows}(M)\end{aligned}$$

The first rule describes both asymmetric encryption and signing (when K is a private signing key). The second rule expresses that the intruder can decrypt an encrypted message when he knows the corresponding private key (denoted by $\mathsf{inv}(\cdot)$), and the third rule expresses that one can always obtain the text of a digital signature (the verification of signatures is expressed in transition rules of honest agents using pattern matching). The last two rules describe symmetric encryption and decryption, respectively.

We may have further similar rules for intruder deduction. As is standard, we assume that a subset of all function symbols are *public*, such as encryption, concatenation, public-key tables, etc. The intruder can use these symbols to form new messages, namely, for each public symbol f of arity n, we have the rule:

$$\mathsf{iknows}(M_1).\cdots.\mathsf{iknows}(M_n) \Rightarrow \mathsf{iknows}(f(M_1,\ldots,M_n)) \ .$$

We assume that all constants that represent agent names and public keys are public symbols (of arity 0). We may also consider algebraic properties such as $\exp(\exp(g, X), Y) \approx \exp(g, Y), X)$ that we need for the Diffie-Hellman key exchange. While we allow for algebraic properties in general, for the results we are interested in here we assume that the symbols $\{\cdot\}.$, $\{|\cdot|\}.$, and $\cdot, \cdot$ (for pairing) that we use in our model do not have algebraic properties.

We consider here a Dolev-Yao-style intruder model, in which the intruder controls the network as explained above, including that he can send messages under an arbitrary identity. Moreover, he may act, under his real name, as a normal agent in protocol runs. We generalize this slightly and allow the intruder to have more than one "real name", i.e. he may have several names that he controls, in the sense that he has the necessary long-term keys to actually work under a particular name. This reflects a large number of situations, like an honest agent who has been compromised and whose long-term keys have been learned by the intruder, or when there are several dishonest agents who all collaborate. This worst case of a collaboration of all dishonest agents is simply modeled by one intruder who acts under different identities. To that end, we use the fact symbol $\mathsf{dishonest}(A)$ that holds true for every dishonest agents A (from the initial state on). We can also allow for IF rules that model the compromise of an agent A by giving the intruder all knowledge of A and adding the fact $\mathsf{dishonest}(A)$. We will use this also for pseudonyms freshly created by the intruder for pseudonymous channels. More specifically, to ensure that the intruder can generate himself new pseudonyms at any time and can send and receive messages with these new pseudonyms, we use the predicate $\mathsf{dishonest}(\cdot)$ in the rule:

$$=[\psi]\Rightarrow \mathsf{iknows}(\psi).\mathsf{iknows}(\mathsf{inv}(\psi)).\mathsf{dishonest}(\psi)\,.$$

This includes $\mathsf{inv}(\psi)$, which we need for the CCM, where pseudonyms are simply public keys (as, e.g., in PBK). Creating a new pseudonym thus means generating a key pair $(\psi, \mathsf{inv}(\psi))$.

Attack states are formalized in IF by means of the attack rules in G, which are rules without a right-hand side: a state at which an attack rule $L \mid \mathit{cond}$ can fire is thus an attack state.

3 Channels as Assumptions

We now define two formal models for channels as assumptions, summarized in Table 1: the *ideal channel model ICM* describes the properties of a channel in an ideal way using IF facts, while the *cryptographic channel model CCM* employs cryptography to achieve the same properties on the basis of insecure channels. We will also show that the CCM implements the ICM in a certain sense.

3.1 The Ideal Channel Model ICM

We introduce new facts $\mathsf{athCh}_{A,B}(M)$, $\mathsf{cnfCh}_B(M)$ and $\mathsf{secCh}_{A,B}(M)$ to express that an incoming or outgoing message is transmitted on a particular kind of

Table 1. Channels as assumptions in the ICM and the CCM.

Channel	AnB•	ICM	CCM
Insecure	$A \rightarrow B : M$	$\mathsf{iknows}(M)$	$\mathsf{iknows}(M)$
Authentic	$A \bullet\!\!\rightarrow B : M$	$\mathsf{athCh}_{A,B}(M)$	$\mathsf{iknows}(\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))})$
Confidential	$A \rightarrow\!\!\bullet\, B : M$	$\mathsf{cnfCh}_B(M)$	$\mathsf{iknows}(\{\mathsf{ctag}, M\}_{\mathsf{ck}(B)})$
Secure	$A \bullet\!\!\rightarrow\!\!\bullet\, B : M$	$\mathsf{secCh}_{A,B}(M)$	$\mathsf{iknows}(\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)})$

channel where A and B can be either real names or pseudonyms and M is the transmitted message. We refer to these three facts as *ICM facts* or *channel facts*. In contrast to the insecure channels, the authentic and secure channels also have sender and receiver names, and the confidential channels only the receiver names, as this information is relevant for their definition. Also, like for the $\mathsf{iknows}(\cdot)$ facts, we define the $\mathsf{athCh}_{A,B}(M)$, $\mathsf{cnfCh}_B(M)$ and $\mathsf{secCh}_{A,B}(M)$ facts as persistent. Thus, once a message is sent on any of these channels, it "stays there" and can be received an arbitrary number of times by any receiver. Therefore, these channels do not include a freshness guarantee or protection against replay; we discuss such a channel variant in [26]. Finally, we require that the channel facts do not occur in the initial state or the goals. Then, for instance, the second transition of A for our example protocol P of Fig. 1 looks as follows (cf. (1)):

$$\begin{aligned}\mathsf{state}_{\mathcal{A}}(A, 1, B, g, X).\mathsf{athCh}_{B,A}(GY) \dashv[Payload]\Rightarrow\\ \mathsf{iknows}(\{|Payload|\}_{\exp(GY,X)}).\mathsf{state}_{\mathcal{A}}(A, 2, B, g, X, GY, Payload)\end{aligned} \tag{2}$$

A thus processes the incoming message only if there is a message on an authentic channel such that B and A match the respective values in the local state of A. Due to persistence, the left-hand side fact $\mathsf{athCh}_{B,A}(GY)$ is not removed by applying this rule.

With this, we have already defined part of the properties of the channels implicitly, namely the behavior of honest agents for channels: they can send and receive messages as described by the transition rules. In particular, since we have defined channel facts to be persistent, an agent can receive a single message on such a channel any number of times. What is left to define is the intruder behavior. This is defined by the rules in Fig. 2 that define the abilities of the intruder on these channels and thus their ideal functionality:

(3) He can send messages on an authentic channel only under the name of a dishonest agent A to any agent B.
(4) He can receive any message on an authentic channel.
(5) He can send messages on a confidential channel to any agent B.
(6) He can receive messages on a confidential channel only when they are addressed to a dishonest agent B.
(7) He can send messages on a secure channel to any agent B but only under the name of a dishonest agent A.
(8) He can receive messages on a secure channel whenever the messages are addressed to a dishonest agent B.

Note that all occurrences of "only" in these explanations are due to the fact that we do not describe further rules for the intruder that deal with the channels.

$$\mathsf{iknows}(B).\mathsf{iknows}(M).\mathsf{dishonest}(A) \Rightarrow \mathsf{athCh}_{A,B}(M) \quad (3)$$
$$\mathsf{athCh}_{A,B}(M) \Rightarrow \mathsf{iknows}(M) \quad (4)$$
$$\mathsf{iknows}(B).\mathsf{iknows}(M) \Rightarrow \mathsf{cnfCh}_B(M) \quad (5)$$
$$\mathsf{cnfCh}_B(M).\mathsf{dishonest}(B) \Rightarrow \mathsf{iknows}(M) \quad (6)$$
$$\mathsf{iknows}(B).\mathsf{iknows}(M).\mathsf{dishonest}(A) \Rightarrow \mathsf{secCh}_{A,B}(M) \quad (7)$$
$$\mathsf{secCh}_{A,B}(M).\mathsf{dishonest}(B) \Rightarrow \mathsf{iknows}(M) \quad (8)$$

Fig. 2. The intruder rules for the ICM.

3.2 The Cryptographic Channel Model CCM

We have now defined channels in an abstract way by their ideal behavior. This behavior can be realized in a number of different ways, including non-electronic implementations, such as sealed envelopes or a face-to-face meetings of friends. The CCM that we present now is one possible cryptographic realization based on asymmetric cryptography. We first consider the case of agents identified by their real names. For this model, we introduce new symbols atag, ctag, stag, ak and ck. Here, atag, ctag, and stag are tags to distinguish the channel types, while ak and ck are tables of public keys, for signing and encrypting, respectively. Thus, $\mathsf{ak}(A)$ and $\mathsf{ck}(A)$ are the public keys of agent A, and $\mathsf{inv}(\mathsf{ak}(A))$ and $\mathsf{inv}(\mathsf{ck}(A))$ are the corresponding private keys. We refer to all these keys and tags as *CCM material*. We assume that every agent, including the intruder, knows initially both keytables ak and ck and its own private keys. Thus the additional initial intruder knowledge of the CCM is

$$\{\mathsf{ak}, \mathsf{ck}, \mathsf{atag}, \mathsf{ctag}, \mathsf{stag}\} \bigcup_{\mathsf{dishonest}(A)} \mathsf{inv}(\mathsf{ak}(A)) \bigcup_{\mathsf{dishonest}(A)} \mathsf{inv}(\mathsf{ck}(A)) \,. \quad (9)$$

For the rules of honest agents, we express incoming and outgoing messages as described in Table 1. For instance, the second transition of A for our example of Fig. 1 looks as follows (cf. (1) and (2) in the ICM):

$$\mathsf{state}_{\mathcal{A}}(A, 1, B, g, X).\mathsf{iknows}(\{\mathsf{atag}, A, GY\}_{\mathsf{inv}(\mathsf{ak}(B))}) =\![Payload]\!\Rightarrow \\ \mathsf{iknows}(\{\!|Payload|\!\}_{\exp(GY,X)}).\mathsf{state}_{\mathcal{A}}(A, 2, B, g, X, GY, Payload)$$

A thus processes the incoming message only if it correctly encodes an authentic message from B for A according to the CCM definition.

Observe that for the authentic and secure channels, we include the name of the intended recipient in the signed part of the message. This inclusion ensures that a message cannot be redirected to a different receiver. To see that, consider the alternative encoding of a secure channel (and similarly for the authentic channel) that does not include the name: $\{\{\mathsf{stag}, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$. If B is dishonest, he can decrypt the outer encryption to obtain $\{\mathsf{stag}, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ and re-encrypt it for any other agent C, i.e. $\{\{\mathsf{stag}, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(C)}$. This message would erroneously appear as one from A for C. Such a mistake was indeed often a source of problems in security protocols, e.g. [10]. Such attacks are prevented by

our construction to include the receiver name in the signed part of the message. For an authentic channel, this corresponds to our previous observation that the channel should also include the authentic transmission of the intended receiver name. This also ensures that a secure channel combines the properties of an authentic and a confidential channel.

To integrate pseudonymous agents into the CCM, i.e. to implement cryptographically pseudonyms that can serve as a basis for secure channels, we employ the popular idea (see e.g. [7]) of using a public key (or a hash of a public key) as a pseudonym and define ownership of such a pseudonym by knowledge of the corresponding private key. Thus, every agent, including the intruder, can create any number of pseudonyms, and, assuming private keys are never revealed, the "theft" of pseudonyms is impossible. The encoding of the different channel types is now the same as in the case of real names, except that instead of the keys $\mathsf{ak}(A)$ and $\mathsf{ck}(A)$ related to the real name, we directly use the pseudonym. For instance, sending a message M on a confidential channel to an agent under pseudonym ψ is simply encoded by $\{\mathsf{ctag}, M\}_\psi$.

3.3 Relating the two Channel Models

We now show that we can simulate in a certain sense every behavior of the ICM also in the CCM. This means that it is safe to verify protocols in the CCM since every attack in the ICM has a counter-part in the CCM. A simulation in the other direction is possible under some further assumptions related to typing. The two directions of the simulation together show that the two models are in some sense equivalent, in particular that the cryptographic channels correctly implement ideal channels. This result guarantees that we do not have any false positives with respect to the ICM, i.e. attacks that only work in the CCM.

It should be intuitively clear what we mean when we talk about, for instance, an *ICM protocol specification and the corresponding CCM specification* or *corresponding states* in such models. However, to formally prove anything about such corresponding specifications, we need to define the notions:

Definition 1. *Consider two IF specifications $P_1 = (I, R_1, G)$ and $P_2 = (I', R_2, G)$, where I is an initial state that contains no ICM channel facts and no CCM material, I' is I augmented with the knowledge of (9), G is a set of goals that does not refer to ICM channel facts and CCM material, and R_1 and R_2 are sets of rules for honest agents where*

- *the rules of R_1 contain no CCM material,*
- *the rules of R_2 contain no ICM channel facts,*
- *and $f(R_1) = R_2$ for a translation function f that replaces every ICM channel fact that occurs in the rules of R_1 with the corresponding intruder knowledge of the CCM.*

We then say that P_1 is an ICM specification *and P_2 is a* CCM specification*, and that P_1 and P_2* correspond *to each other. We define an equivalence relation $\sim$ for states S_i: we have $S_1 \sim S_2$ iff*

– *S_1 and S_2 contain the same facts besides ICM facts and* $\mathsf{iknows}(\cdot)$ *facts,*
– *the intruder knowledge in S_1 and S_2 is the same when removing all messages that contain CCM material, and*
– *the channel facts and intruder knowledge of crypto-encodings are equivalent in both states modulo the mapping in Table 1.*

Theorem 1. *Consider an ICM specification and the corresponding CCM specification, both employing real names and/or pseudonyms. For a reachable state S_1 of the ICM specification, there is a reachable state S_2 of the CCM specification such that $S_1 \sim S_2$.*

As we remarked, the proofs of all our theorems can be found in [26]. To establish the converse direction, we need two additional assumptions (which are are sufficient for Theorem 2 but not necessary). First, we need a *typed model*, where every message term has a unique type. There are several *atomic* types such as *nonce*, *publickey*, etc., and we have type constructors for the cryptographic operations, e.g. $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ is of type $\{tag, agent, \tau\}_{privatekey}$ if M is of type τ.

The messages that an honest agent expects according to the protocol are described by a pattern (i.e. a message with variables) and this pattern has a unique type. This does not ensure, however, that the agent accepts only correctly typed messages, i.e. the intruder can send ill-typed messages. For many protocols one can ensure, e.g. by a tagging scheme, that every ill-typed attack can be simulated by a well-typed one [19], so one can focus on well-typed attacks without loss of generality. We will not prescribe any particular mechanism here, but simply assume a well-typed attack.

The second assumption is that a message can be *fully analyzed* by an honest receiver in the sense that its message pattern contains only variables of an atomic type. This means for instance, that we exclude (in the following theorem) protocols like Kerberos where A sends to B a message encrypted with a shared key K_{AC} between A and C, where B does not know K_{AC} and so B cannot decrypt that part of a message. Therefore, the message pattern of B would contain a variable of type $\{\!|\cdot|\!\}_{\cdot}$ which is not atomic. When all its messages can be fully analyzed by honest receivers, then we say that a protocol specification is with *full receiver decryption.*

Theorem 2. *Consider an ICM specification and the corresponding CCM specification, both employing real names and/or pseudonyms and both with full receiver decryption, and consider a well-typed attack on the CCM specification that leads to the attack state S_2. Then there is a reachable attack state S_1 of the ICM specification such that $S_1 \sim S_2$.*

Theorems 1 and 2 relate the ICM and the CCM by showing that attacks in either model can be simulated in the other. On the theoretical side, relating ideal functionality and cryptographic implementation gives us insight in the meaning of channels as assumptions. On the practical side, it allows us to use both models interchangeably in protocol analysis tools that may have different preferences.

4 Channels as Goals

We now specify goals of a protocol using the different kinds of channels. Intuitively, this means that the protocol should ensure the authentic, confidential, or secure transmission of the respective message. These definitions are close to standard ones of security protocols, e.g. [5,21,24].

In order to formulate the goals in a protocol-independent way, we use a set of *auxiliary events* of the protocol execution as an interface between the concrete protocol and the general goals. The use of such auxiliary events is common to IF and several other approaches (e.g. Casper [22]). In addition to the standard auxiliary events $\mathsf{witness}(\cdot)$ and $\mathsf{request}(\cdot)$ of IF, we consider here the events $\mathsf{whisper}(\cdot)$ and $\mathsf{hear}(\cdot)$. These auxiliary events express information about honest agents' assumptions or intentions when executing a protocol: they provide a language over which we then define protocol properties and they are, in general, added to the protocol description by the protocol modeler at specification time. The intruder can neither generate auxiliary events nor modify those events generated by honest agents.

For simplicity, we assume for a goal of the form

$$A \textit{ channel } B : M$$

that M is atomic and freshly generated by A during the protocol in a uniquely determined rule r_A. Similarly, we assume that there is a uniquely determined rule r_B where the message M is learned by B. (If there is no such rule where B learns the message, then the goal is not meaningful.) This allows for protocols where M is not directly sent from A to B, and for protocols where B receives a message that contains M as a subterm, but from which B cannot learn M yet.

For the goal $A \bullet\!\!\rightarrow B : M$, we add the fact $\mathsf{witness}(A, B, P, M)$ to the right-hand side of r_A and the fact $\mathsf{request}(A, B, P, M)$ to the right-hand side of r_B; here, P is an identifier for the protocol.[1] For the goal $A \rightarrow\!\!\bullet\, B : M$, we add the fact $\mathsf{whisper}(B, P, M)$ to the right-hand side of r_A and the fact $\mathsf{hear}(B, P, M)$ to the right-hand side of r_B. For the goal $A \bullet\!\!\rightarrow\!\!\bullet\, B : M$, we add both the facts of authentic and confidential channels to r_A and r_B, respectively.

Intuitively, the additional facts for r_A express the intention of A to send M to B on the respective kind of channel, and the fact for r_B expresses that B believes to have received M (from A in a $\mathsf{request}(\cdot)$ fact for an authentic channel, and from an unspecified agent in a $\mathsf{hear}(\cdot)$ fact for a confidential channel) on the respective kind of channel.

When the goal is a confidential or secure channel, then M must be confidential from its creation on; otherwise there can be trivial attacks. This excludes

[1] One may consider a variant where the P is replaced by a unique identifier for the protocol variable M so to distinguish implicitly several channels from A to B. (In fact, this is standard in authentication goals, distinguishing the interpretation of data.) This identifier has then to be included in the ICM and CCM as well to achieve the compositionality result below. We have chosen not to bind an interpretation to the messages sent on the channels in this paper but note that the results are similar, mutatis mutandum.

$$\mathsf{request}(A,B,P,M) \mid \mathsf{not}(\mathsf{witness}(A,B,P,M)).\mathsf{not}(\mathsf{dishonest}(A)) \quad (10)$$

$$\mathsf{request}(A,B,P,M).\mathsf{dishonest}(A) \mid \mathsf{not}(\mathsf{iknows}(M)) \quad (11)$$

$$\mathsf{whisper}(B,P,M).\mathsf{iknows}(M) \mid \mathsf{not}(\mathsf{dishonest}(B)) \quad (12)$$

$$\mathsf{hear}(B,P,M) \mid \mathsf{not}(\mathsf{whisper}(B,P,M)).\mathsf{not}(\mathsf{iknows}(M)) \quad (13)$$

Fig. 3. Attack states for defining channels as goals.

some protocols (as insecure), namely those that first disclose M to an unauthenticated agent, and consider M as a secret only after authenticating that agent. Such protocols are however not suitable for implementing confidential or secure channels anyway, while they may be fine for, e.g., a key exchange.

We can now define attacks in a protocol-independent way based on the attack states in Fig. 3. The rules (12) and (10) reflect the standard definition of secrecy and authentication goals (non-injective agreement in the terminology of [21]; we consider the injective variant in [26]). For authentic messages, a violation occurs when an honest agent B — B must be honest since the intruder never creates any $\mathsf{request}(\cdot)$ facts — accepts a message as coming from an honest agent A but A has never said it. That is, $\mathsf{request}(A,B,P,M)$ holds but neither $\mathsf{witness}(A,B,P,M)$ nor $\mathsf{dishonest}(A)$ hold. For confidential messages, a violation occurs when M was sent by an honest agent A — since $\mathsf{whisper}(\cdot)$ is never generated by the intruder — for an honest agent B and the intruder knows M. Note that with respect to the standard definitions of goals, we have generalized the notion of the intruder name to arbitrary identities controlled by the intruder (in accordance to what we said about the intruder model in § 3.1).

Additionally, we have the two goals (11) and (13) that are usually not considered in protocol verification, and that we found missing when proving the compositionality result in § 5. These concern the cases when an intruder is the sender of an authentic or confidential message. In these cases, the intruder can of course send whatever he likes, but we consider it as an attack if the intruder is able to convince an agent that he authentically or confidentially said a particular message when in fact he does not know this message. To illustrate this, consider the simple protocol

$$A \to B : \{M\}_{k(B)}, \{h(M)\}_{\mathsf{inv}(k(A))}$$

with the goal $A \bullet\!\!\rightarrow B : M$. A dishonest i can intercept such a message and send to B the modified message $\{M\}_{k(B)}, \{h(M)\}_{\mathsf{inv}(k(i))}$, thereby acting as if he had said M, even though he does not know it. For the classical authentication goals, this is not a violation, but our attack rule (11) matches with this situation. We count this as a flaw since sending a message that one does not know on an authentic channel is not a possible behavior of the ideal channel model.

5 Compositional Reasoning for Channels

We now show that, under certain conditions, a protocol providing a particular channel as goal can be used to implement a channel that another protocol assumes (in the ICM). This composition problem is related to many other problems, such as running several protocols in parallel. There is a variety of literature on this, offering different sets of sufficient conditions for such a parallel composition, such as using disjoint key-spaces or tagging for the protocol, e.g. [2,11,16,17]. The idea is to disambiguate the interpretation of messages when several protocols use similar message formats, i.e. when there is the danger that (a part of) a message can be interpreted in several different ways. We do not want to commit to particular such composition arguments nor to dive into the complex argumentations behind this.

In fact, in this paper we focus on one particular aspect of compositionality, namely composing protocols assuming channels with protocols realizing them. Thus, we "blank out" other compositionality problems and instead provide an abstract notion of horizontal and vertical composability that does not require a particular composition argument. We then prove that the implementation of a channel by a protocol providing that channel is possible for any protocols that satisfy our composability notion.

We first consider the *horizontal* composition of protocols, running different protocols in parallel (as it is standard, see, for instance, [2,11,16,17]), in contrast to using one protocol over a channel provided by another.

Definition 2. *Let Π be a set of protocols and P be a protocol. We denote with $Par(P)$ the system that results from an unbounded number of parallel executions of P, and with $\|_{P\in\Pi}\ Par(P)$ the system that results from running an unbounded number of parallel executions of the protocols of Π. We call Π* horizontally composable *if an attack against $\|_{P\in\Pi}\ Par(P)$ implies an attack against $Par(P)$ for some $P \in \Pi$. (Here, an attack against $\|_{P\in\Pi}\ Par(P)$ means that the goal of some $P \in \Pi$ is violated.)*

Trivially, a set of protocols is horizontally composable iff any of them has an attack. To see that this definition is indeed useful, consider a set of protocols for which their individual correctness is not obvious, but may be established by some automated method (which may fail on the composition of the protocols due to the complexity of the resulting problem). The compositionality may however follow from a static argument about the construction of the protocols, such as the use of encryption with keys from disjoint key-spaces. Such an argument in general does not tell us anything about the correctness of the individual protocols, but rather, if they are correct, then so is also their composition.

For our result for reasoning about channels, we need at least that the "lower-level" protocols that implement the different channels are horizontally composable. But we need a further assumption, since we want to use one protocol to implement channels for another. For the rest of this section, we consider only protocol specifications P_1 and P_2 that are given in AnB• notation and where only one transmission over an authentic, confidential, or secure channel in P_2 is

replaced by P_1. A definition on the IF level would be technically complicated (although intuitively clear) and we avoid it here. Multiple uses of channels can be achieved by applying our compositionality theorem several times (given that the protocols are suitable for multiple composition).

Definition 3. *Let P_1 be a protocol that provides a channel $A' \bullet\!\!\rightarrow B' : M'$ as a goal, and P_2 be a protocol that assumes a channel $A \bullet\!\!\rightarrow B : M$ for some protocol message M. Let M' in P_1 be freshly generated by A', and let all protocol variables of P_1 and P_2 be disjoint. We denote by $P_2[P_1]$ the following modification of P_2:*

- *Replace the line $A \bullet\!\!\rightarrow B : M$ with the protocol $P_1\sigma$ under the substitution $\sigma = [A' \mapsto A, B' \mapsto B, M' \mapsto M]$.*
- *Augment the initial knowledge of A in P_2 with the initial knowledge of A' in P_1 under σ and the same for B. Also add the specification of the initial knowledge of all other participants of P_1 (if there are any) to P_2.*

We use the same notation for compositions for confidential and secure channels, where we additionally require that the term M in P_2 contains a nonce that A freshly generates and that does not occur elsewhere in the protocol.

The inclusion of a fresh nonce in the message M of P_2 for confidential and secure channels is needed since otherwise we may get trivial attacks (with respect to P_1) if a confidential or secure channel is used for a message that the intruder already knows (for instance an agent name); since the nonce is fresh, the intruder cannot already know M in its entirety. Note that in our model a message is either known or not known to the intruder, but indistinguishability is not considered. The simple inclusion of some unpredictable element in the payload message implies that the intruder cannot a priori know it.

We now define the *vertical* composition of protocols P_1 and P_2. Intuitively, it means that P_1 and P_2 are composable in the previous, horizontal sense, when using arbitrary messages from P_2 in place of the payload-nonce in P_1.

Definition 4. *Let P_1 and P_2 be as in Definition 3. For every honest agent A and every agent B, let $\mathcal{M}_{A,B}$ denote the set of concrete payload messages (i.e. instances of M) that A sends in any run of P_2 to agent B.*[2] *Let P_1^* be the variant of protocol P_1 where in each run each honest agent A chooses the payload message M' arbitrarily from $\mathcal{M}_{A,B}$ instead of a freshly generated value. We say that P_2 is* vertically composable *with P_1, if P_1^* and P_2 are horizontally composable.*

With this, we have set out two challenging problems: a verification problem and a horizontal composition problem where one of the protocols, P_1^*, uses payload messages from an, in general, infinite universe. We do not consider how to solve these problems here, and merely propose that under some reasonable assumptions these problems can be solved. In particular, we need to ensure that the messages and submessages of the protocols cannot be confused and that the behavior of P_1^* is independent from the concrete payload message, e.g. by using

[2] Assuming that the fresh data included in payload messages is taken from pairwise disjoint sets $X_{A,B}$ (which is not a restriction) then also the $\mathcal{M}_{A,B}$ are disjoint.

tagging. Under certain conditions, we may then verify P_1 with a fresh constant as a "black-box payload message" instead of P_1^*.

Theorem 3. *Consider protocols P_1, P_1^*, and P_2 as in Definition 4 where endpoints may be pseudonymous, and let P_1 and P_2 be vertically and horizontally composable. If there is no attack against P_1, P_1^*, and P_2, then there is no attack against $P_2[P_1]$.*

Example 1. As a simple illustration of the application and strength of this result, let us return to our running example and consider an attack that results from protocol composition; this attack is relatively trivial but it suffices to illustrate the main points. Consider as P_2 our example protocol P of Fig. 1 and let us implement the first authentic channel by the protocol P_1 below on the left. The composition $P_2[P_1]$ is shown on the right.

$$\frac{A' \;\to\; B' : \{B', M'\}_{\mathsf{inv}(\mathsf{pk}(A'))}}{A' \;\bullet\!\!\to\; B' : M'} \qquad \frac{\begin{array}{rcl} A & \to & B : \{B, \exp(g, X)\}_{\mathsf{inv}(\mathsf{pk}(A))} \\ B & \bullet\!\!\to & A : \exp(g, Y) \\ A & \to & B : \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \end{array}}{A \;\bullet\!\!\to\!\bullet\; B : Payload}$$

The set of values for the payload $M = \exp(g, X)$ from A to B is $\mathcal{M}_{A,B} = \{g^x \mid x \in X_{A,B}\}$ where $X_{A,B}$ is a countable set of exponents used by A for B such that $X_{a,b} \cap X_{a',b'} = \emptyset$ unless $a = a'$ and $b = b'$. We sketch a proof that P_1^* and P_2 are horizontally composable. Recall that this does not require that P_1^* or P_2 themselves are correct, but that their combination cannot give an attack against either protocol that would not have worked similarly on that protocol in isolation. First, observe that the signed messages of P_1^* are not helpful to attack P_2 (because P_2 does not deal with signatures and the intruder may instead use any other message as well). Second, the content of the signed messages in P_1^* are the half-keys from P_2, i.e. the intruder can learn each such message in a suitable run of P_2. Vice-versa, P_2 is not helpful to attack P_1^*, since P_2 does not deal with signatures, so he can only introduce message parts from P_2 that he signed himself (under any dishonest identity) and since he must know such messages, this cannot give an attack against P_1^*.

Consider the following variant P_2':

$$\frac{\begin{array}{rcl} A & \to & B : \{B, G\}_{\mathsf{inv}(\mathsf{pk}(A))} \\ A & \bullet\!\!\to & B : \exp(G, X) \\ B & \bullet\!\!\to & A : \exp(G, Y) \\ A & \to & B : \{\!|Payload|\!\}_{\exp(\exp(G,X),Y)} \end{array}}{A \;\bullet\!\!\to\!\bullet\; B : Payload}$$

This is a variant of the Diffie-Hellman key exchange, which we intentionally designed so that it breaks when composing it with P_1. In the additional first message, A authentically transmits a basis G that she chooses for the key exchange. While P_2' is also correct in isolation, running P_2' and P_1^* in parallel leads

to an attack since the first message of P_2' has the same format as the message of P_1^*; namely, when an agent a sends the first message of P_2'

$$a \rightarrow b : \{b, g\}_{\mathsf{inv}(\mathsf{pk}(a))}$$

it may be falsely interpreted by b as P_1, leading to the event $\mathsf{request}(a, b, p_1, g)$ for which no corresponding witness fact exists (since a did not mean it as P_1). Thus, there is a trivial authentication attack.

6 Related Work and Conclusions

We conclude by discussing relevant related works and pointing to directions for future research, in addition to those that we already mentioned above.

In [23], Maurer and Schmid introduce the $\bullet$ notation to give a calculus for reasoning about what new (authentic, confidential, secure) channels can be built from given ones, but the notation is never directly used for transmitting messages (although the informal arguments consider concrete message transmissions). Since they do not formally define their channels, it is hard to tell from the way they intuitively explain and use the notation how their understanding of channels relates to ours, but it seems to be closest to the fresh variants of the channels that we discuss in [26], where we formalize the extension of the channels considered here to prevent the replay of messages.

Dilloway and Lowe [15] consider the specification of secure channels, used as assumptions, in a formal/black-box cryptographic model. They define several channel types similar to our standard channel types with real names, but they include also some weaker types of channels that we did not consider because the respective stronger channels come at little extra cost (like including the intended recipient on an authentic channel).

Like [15], Armando et al. in [4] characterize channels as assumptions by restricting the traces that are allowed for the different channel types, in contrast to our "constructive" approach of describing explicitly what agents can do. While they do not consider all the channel types in their work, they can model resilient channels by excluding traces where sent messages are never received.

In [1], Abadi et al. give a general recipe for constructing secure channels, albeit with a notion different from all the above works: their goal is to construct a channel such that a distributed system based on this channel should be indistinguishable for an attacker from a system that uses internal communication instead. This is a much stronger notion of channels than ours, and one that is more closely related to the system that uses them. It is, of course, more expensive to achieve this notion. For instance, all messages are repeatedly sent over the channel to avoid that an intruder blocking some messages of the channel can detect a difference in the behavior of the system. [8] considers a similar approach.

Much effort has been devoted to protocol composition in the formal verification area, e.g. [2,11,12,13,16,17]. As we remarked, different sets of sufficient conditions (such as using disjoint key-spaces or tagging for the protocol) have been formalized for the horizontal compositionality problem that results from

running several protocols in parallel. A particular challenge arises when the composed protocols are not unrelated (and one has to merely prevent interactions) but are rather related sub-protocols of a larger system as in [16,17]. While we have considered a different kind of problem with our vertical composition result, i.e. running one protocol "on top of another", the problems and assumptions we rely on are related. For Theorem 3, in particular, we have assumed the verification of P_1^*, i.e. the transmission protocol inserting an arbitrary payload message (from a certain set). We are currently investigating how this can be done without considering the concrete payloads in the verification of P_1; the hope is that we can employ meta-arguments based on some structural properties of the protocols similar to said compositionality results.

There are two frameworks for the secure composition of cryptographic primitives and protocols: *Universal Composability* [9] and *Reactive Simulatability* [6]. Both stem from the cryptographic world, and are based on the notion that the implementation of an ideal system is secure if no computationally limited attacker with appropriate interfaces to both the ideal system and the implementation can distinguish them. The view of cryptography through indistinguishability from an ideal system is not directly feasible for the automated verification of security protocols. All the arguments in this paper are within a black-box cryptography world and have not been related to cryptographic soundness. Even though for many applications such models are indeed cryptographically sound [27], the transition from a cryptographic model to a black-box model in general implies the exclusion of (realistic) attacks. The simulation proofs between black-box cryptography models (as in all our theorems) show that we do not loose *further* attacks by considering simpler verification problems or models that are better suited for a particular verification technique. Thus, once committed to a black-box model, we can safely simplify the automated verification by exploiting our theorems. Besides this, the simulation also gives us insights in the properties of our formal models and we plan to investigate the relation of such results in the formal world with the cryptographic world as future work.

Acknowledgments The work presented in this paper was partially supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" and the PRIN'07 project "SOFT". We thank Thomas Gross, Birgit Pfitzmann and Patrick Schaller.

References

1. M. Abadi, C. Fournet, and G. Gonthier. Secure Implementation of Channel Abstractions. *Information and Computation*, 174(1):37–83, 2002.
2. S. Andova, C. Cremers, K. Gjøsteen, S. Mauw, S. Mjølsnes, and S. Radomirović. A framework for compositional verification of security protocols. *Information and Computation*, 206:425–459, 2008.
3. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool

for the Automated Validation of Internet Security Protocols and Applications. In *Proc. CAV'05*, LNCS 3576:281–285. Springer, 2005.
4. A. Armando, R. Carbone, and L. Compagna. LTL Model Checking for Security Protocols. In *Proc. CSFW'07*:385–396. IEEE CS Press, 2007.
5. AVISPA. Deliverable 2.3: The Intermediate Format. Available at `www.avispa-project.org`, 2003.
6. M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems, 2004. Cryptology ePrint Archive, Report 2004/082, `http://eprint.iacr.org/`
7. S. Bradner, A. Mankin, and J. Schiller. A framework for purpose built keys (PBK), 2003. Work in Progress (Internet Draft: `draft-bradner-pbk-frame-06.txt`).
8. M. Bugliesi and R. Focardi. Language based secure communication. In *Proc. CSFW'08*:3–16. IEEE CS Press, 2008.
9. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. FOCS'01*:136–145. IEEE CS Press, 2001.
10. I. Cervesato, A. D. Jaggard, A. Scedrov, J.-K. Tsay, and C. Walstad. Breaking and fixing public-key Kerberos. *Information and Computation*, 206:402–424, 2008.
11. V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, 2009.
12. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proc. FMSE'03*:11–23. ACM Press, 2003.
13. S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *Proc. CSFW'08*:239–251. IEEE CS Press, 2008.
14. T. Dierks and C. Allen. RFC2246 – The TLS Protocol Version 1, 1999.
15. C. Dilloway and G. Lowe. On the specification of secure channels. In *Proc. WITS '07*, 2007.
16. J. D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *J. Comp. Sec.*, 3–4(12):409–433, 2004.
17. J. D. Guttman. Cryptographic protocol composition via the authentication tests. In *Proc. FOSSACS'09*, LNCS 5504:303–317. Springer, 2009.
18. P. Hankes Drielsma, S. Mödersheim, L. Viganò, and D. Basin. Formalizing and analyzing sender invariance. In *Proc. FAST'06*, LNCS 4691:80–95. Springer, 2007.
19. J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proc. CSFW'00*:217–244. IEEE CS Press, 2000.
20. D. Johnson, C. Perkins, and J. Arkko. RFC3775–Mobility Support in IPv6, 2004.
21. G. Lowe. A hierarchy of authentication specifications. In *Proc. CSFW'97*:31–43. IEEE CS Press, 1997.
22. G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. *J. Comp. Sec.*, 6(1):53–84, 1998.
23. U. M. Maurer and P. E. Schmid. A calculus for security bootstrapping in distributed systems. *J. Comp. Sec.*, 4(1):55–80, 1996.
24. S. Mödersheim. *Models and Methods for the Automated Analysis of Security Protocols*. PhD Thesis, ETH Zurich, 2007. ETH Dissertation No. 17013.
25. S. Mödersheim. Algebraic Properties in Alice and Bob Notation. In *Proc. Ares'09*, Full version: T. Rep. RZ3709, IBM Zurich Research Lab, 2008, `domino.research.ibm.com/library/cyberdig.nsf`.
26. S. Mödersheim and L. Viganò. Secure Pseudonymous Channels (extended version). T. Rep. RZ3724, IBM Zurich Research Lab, 2009. `domino.research.ibm.com/library/cyberdig.nsf`
27. C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner. Cryptographically Sound Theorem Proving. In *Proc. CSFW'06*:153–166. IEEE CS Press, 2006.