RZ 3724 (# 99734) 01/05/2009 (updated 08/03/2009)
Computer Science   22 pages

Research Report

Secure Pseudonymous Channels

(Updated version of August 3, 2009)

Sebastian Mödersheim

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon, Switzerland
E-mail: smo@zurich.ibm.com

Luca Viganò

Department of Computer Science
University of Verona
Verona, Italy
E-mail: luca.vigano@univr.it

Secure Pseudonymous Channels

Sebastian Mödersheim[1] Luca Viganò[2]

[1] IBM Zurich Research Laboratory, Switzerland, smo@zurich.ibm.com

[2] Department of Computer Science, Verona, Italy, luca.vigano@univr.it

Abstract

Different kinds of channels can be employed in security protocols and web services as a means of securing the communication. We consider here three basic kinds of channels: authentic, confidential, and secure.

We define channels as assumptions*, i.e. when the messages of a protocol may be transmitted over such channels. More specifically, we define the* Ideal Channel Model*, which describes the ideal functionality of a channel, and the* Cryptographic Channel Model*, which employs concrete cryptographic messages on insecure channels. We relate these two models by showing that attacks in either model can be simulated in the other.*

We also define the meaning of channels as goals*, i.e. when a protocol has the goal of 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.*

Finally, we generalize all our results to channels where agents may be identified by pseudonyms rather than by their real names, and also consider channels that ensure the freshness of messages by suppressing message replay.

1. Introduction

Context and motivation. In recent years, a number of works have appeared that provide formal definitions of the notion of channel and how different kinds of channels (e.g. authentic, confidential, secure) 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 [28] to the investigation of a lattice of different channel types [20], [21].

In this paper, we consider three basic kinds of channels: authentic, confidential, and secure channels. We use an intuitive notation from [28], where a secure end-point of a channel is marked by a bullet with the following informal meaning (that we define precisely below):

- *Authentic channel*: $A \bullet\!\!\rightarrow B : M$ represents an authentic channel from A to B. This means that B can rely that A has sent the message M.
- *Confidential channel*: $A \rightarrow\!\!\bullet B : M$. This means that A can rely that only B can receive the message M.
- *Secure Channel*: $A \bullet\!\!\rightarrow\!\!\bullet B : M$. This is a channel that is both authentic and secure.

While [28] 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. the messages of a protocol may be transmitted over such channels. Second, the protocol may have the *goal* of establishing a particular kind of channel.

Contributions. We provide four main contributions. First, for channels as assumptions, we formally 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 formally 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 both models interchangeably in verification 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. 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 compositionality results are 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/SSL, 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 generalize all the above models and theorems to include channels where agents may alternatively be identified by pseudonyms rather than by their real names. A typical example is the channel established by an TLS-connection without a client certificate, where the client is not authenticated but we have kind of a secure channel nonetheless. Such a channel between an unauthenticated client and an authenticated server (if we assume that this authentication works) is then sufficient to run, e.g., a password-based login protocol on it, as follows from our compositionality result. In addition to the practical value of this result we are, again, able to theoretically understand better the concepts, namely relating the notion of sender/receiver invariance as goals [22] with the ideal functionality as assumption and its possible cryptographic realization.

Fourth, and last, we consider a variant of channels that ensure freshness of messages, i.e. suppress the replay of messages.

Organization. In § 2 and § 3, we briefly describe the formal languages that we use in this paper. In § 4, we specify standard channels as assumptions and define the ICM and the CCM, and then prove their equivalence. In § 5, we specify standard channels as goals. In § 6, we consider compositional reasoning for standard channels. In § 7, we consider pseudonymous channels, both as assumptions and as goals, and in § 8, we discuss channels that ensure freshness of messages. In § 9, we discuss related work and draw conclusions. To ease the flow, proofs are given in an appendix.

2. A Simple Notation

The definitions and results that 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. We thus introduce a formal language to specify transition systems that employ or provide secure channels. To motivate and illustrate our contributions, let us begin with a limited, but simple and intuitive (and very popular) way to specify security protocols: *Alice and Bob (AnB) notation.*

2.1. Alice and Bob (AnB) notation

While this notation is often used in an informal way, it has provided a basis for several formal specification languages, e.g. [7], [17], [24], [27], [29], [31]. Semantically, a security protocol specification in AnB notation (or an AnB specification, for short) describes a set of processes, one for each protocol role, in terms of the sequence of messages that the role sends and receives when executing the protocol. Such a semantics can be implemented in a compiler that translates an AnB specification into a low-level description suitable for protocol verification tools. For instance, [31] formalizes (and describes a prototype implementation of) a compiler that translates AnB specifications into the *AVISPA Intermediate Format IF* [7], a more low-level (with respect to AnB) language that has itself a clear and well-defined semantics and that is designed to specify transition systems in a way that is feasible for protocol analysis tools. While AnB has a quite limited expressiveness (e.g. we cannot express protocols with repeated parts), IF does not have such limitations and allows one to specify the evolution of arbitrary concurrent processes. In particular, IF is the input language of a variety of tools, most notably the AVISPA Tool itself [3] and its back-ends.

The details of how AnB-style languages work, in particular how honest agents compose, decompose and check messages, is a topic of its own and unrelated to this work on channels. In order to formally define our extensions to AnB without going into all these details, we describe the extension of our own language AnB [31] with the "bullet" (•) notation for channels that we borrow from [28], and thereby use the existing translator/semantics as a black box.

More precisely, we proceed as shown in Figure 1: given an AnB specification that uses our notion of channels, we translate it into one that uses only the syntactic expressions of the existing language (without using the channel bullets) with additional annotations that we will need later on in the translation process. We then use the existing AnB2IF compiler to translate the resulting AnB specification into an IF specification with annotations, and then further transform the IF file exploiting the annotations we introduced on the AnB level. In particular, we define two different ways to encode the channels on the IF side, the Ideal Channel Model ICM and the Cryptographic Channel Model CCM. The result of this whole translation process is then a compiler for our channel extension of AnB.

The different steps of this translation are described in detail for standard channels in the next sections. In § 7 we will then also consider the use of pseudonyms instead of real agent names and modify the translations accordingly. Let us

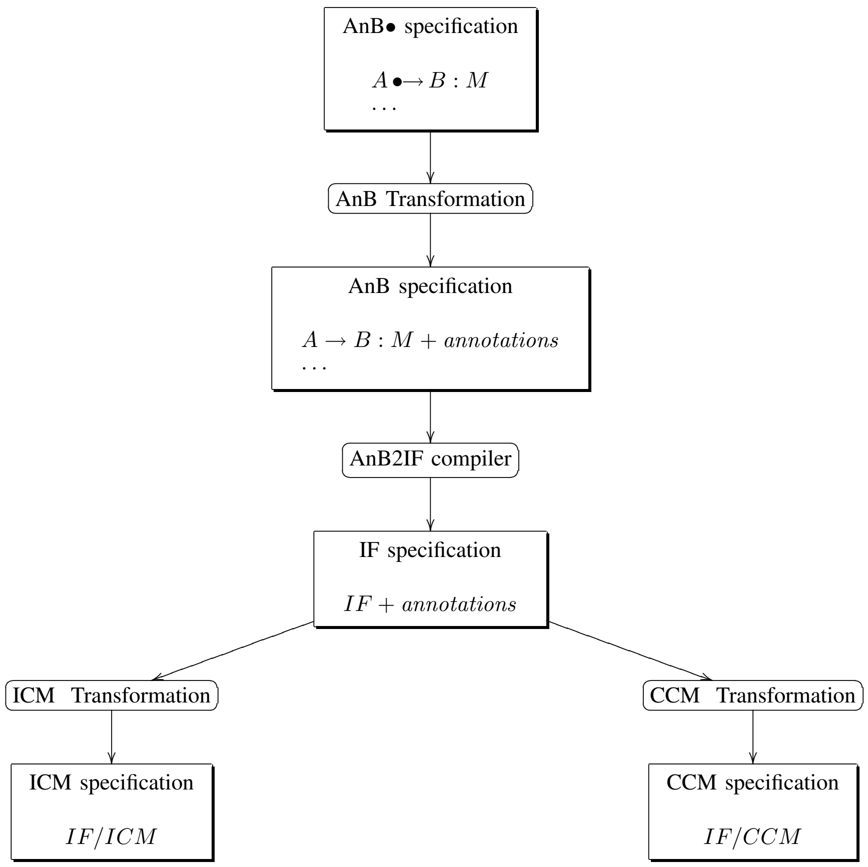

Figure 1. The translations between the models.

begin by considering AnB•, our extension of (specifications written in) the AnB notation with the • annotations for secure (standard) channels borrowed from [28].

2.2. AnB• notation (for standard channels)

As already said, the bullet annotations are used in [28] to reason about the existence of certain channels between agents. In contrast, we here use the extended AnB notation to specify the properties of message exchanges over different kinds of channels, either as an assumption or as a goal. To clarify this, let us begin with the following three kinds of channels, giving informal definitions that we will later make precise:

- *Authentic channel*: $A \bullet\!\!\to B : M$. This means that B can rely that A has sent the message M. Moreover, as we explain in more detail below, our work has revealed the subtle fact that an authentic channel should ensure more, namely that A has meant to say this message to B, i.e. the receiver name is part of what should be authenticated.[1]
- *Confidential channel*: $A \to\!\!\bullet B : M$. This means that A can rely that only B can receive the message M.
- *Secure Channel*: $A \bullet\!\!\to\!\!\bullet B : M$. This is a channel that is both authentic and secure.

The AnB• notation introduces these bullets at two places in specifications written in the AnB notation. First, we can use the channels — as *assumptions* — in the message exchange itself. Intuitively, this means that we assume that the transmission of the respective message has the properties of the channel type (by whatever mechanism this is ensured).

Second, we can use these channels also to specify *goals* of a protocol. Intuitively, this means that the protocol should ensure the authentic, confidential, or secure transmission of the respective message.

As we will see, these goals are similar to standard authentication and secrecy goals. There are however two important differences, which are motivated by our desire to show a composability result of the following form: given a verified protocol P_1 that has a particular kind of channel as a goal, and a verified protocol P_2 that assumes this kind of channel, we can use P_1 to implement the channel in P_2, whenever P_1 and P_2 meet certain criteria. For such a result, the standard authentication and secrecy goals are not sufficient,

1. In fact, the composability results that we prove below based on our definition cannot be achieved without this additional property.

$$\begin{array}{cccl} A & \bullet\!\!\rightarrow & B & : \ \exp(g, X) \\ B & \bullet\!\!\rightarrow & A & : \ \exp(g, Y) \\ A & \rightarrow & B & : \ \{\!|\mathit{Payload}|\!\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : \ \mathit{Payload} \end{array}$$

Figure 2. Example protocol in AnB• notation (excerpt).

and we need a stronger goal. Moreover, authentication and secrecy goals are usually formulated for arbitrary messages transmitted in the protocol, e.g. freshly generated keys, etc. For the composability, we need something more specific: P_1 deals with transmitting a *Payload* message that can be arbitrarily instantiated by a higher-level protocol like P_2 that uses P_1. For such Payload messages, we assume a reserved data type that is left as a black-box in the protocol P_1.

Figure 2 shows an example protocol in AnB• notation (omitting the declaration of types and initial knowledge). This protocol, which we will use as running example, 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 $\{\!|\cdot|\!\}_\cdot$ 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. Note that the translation to IF requires that the translator is aware of the properties of exponentiation. In particular, A will receive some message GY from B (she cannot check that it is indeed a term of the form $\exp(g, Y)$), and she will generate the key as $\exp(GY, X)$ which, for the correct input $GY = \exp(g, Y)$ indeed equals $\exp(\exp(g, X), Y)$. We will continue our example below, giving concrete IF transition rules. For more details on the IF translation, see [31].

2.3. From AnB• to AnB

We now begin with the formalization of the transformations and translations between the different models depicted in Figure 1. Let a protocol specification be given in AnB• notation where channels are used as assumptions and goals. In order to exploit our AnB2IF compiler as a black box, we first transform a specification written in AnB• notation into a specification in AnB notation where messages are annotated in a particular way that reflects the channels (both as assumptions and as goals). This is the step "AnB Transformation" in Figure 1. The resulting AnB specification is then fed into the AnB2IF compiler, which translates it into an IF specification in which messages are annotated analogously. From this annotated IF specification we then finally derive the actual IF specification that corresponds to the given AnB• specification in two variants, the ICM and the CCM. The annotations thus serve as a marking that we "tunnel through" the existing AnB2IF translation process in order to define the extended AnB•2IF translation without opening the AnB2IF black-box.

To that end, we introduce nine new symbols[2] aAnnA, cAnnA, sAnnA, aAnnGS, cAnnGS sAnnGS, aAnnGR, cAnnGR, and sAnnGR. These symbols are used to annotate the messages of the AnB specification for the channels as assumptions (those ending on A), and as goals for the sender (those ending on GS) and the receiver (those ending on GR).

We now define the "AnB Transformation" as follows. For channels as assumptions, the transformation replaces each message exchange in the AnB• notation in the second column of Table 1 with the corresponding annotated message exchange in the AnB notation in the third column of the table, e.g. $A \bullet\!\!\rightarrow B\ :\ M$ is replaced with $A\ \rightarrow\ B\ :$ aAnnA, A, B, M.

For channels as goals, the transformation is more complicated, and we use two annotations for each channel type as shown in the last column of the table. For simplicity, we assume for a goal of the form A *channel* $B : M$ that M is atomic and freshly generated by A during the protocol. We allow that the message is not directly sent from A to B and that it is part of a complex message. Consider the first occurrence of M in the protocol; this must be a message sent by A (because A created it). We annotate this step in the AnB specification with $\mathsf{aAnnGS}, A, B, h(M)$ if the goal is an authentic transmission of M (and analogously for the other channel types). Consider then the step where B first receives M in the sense that B can obtain M from the received message.[3] This may be, but need not be, the same step where A first sends M. We annotate this step with $\mathsf{aAnnGR}, A, B, h(M)$ if the goal is the authentic transmission of M (and analogously for the other channel types). The reason for hashing the message M will become apparent below when we describe the transformation on the IF level.

The result of the "AnB Transformation" step is thus an AnB specification without channels but with annotations that will appear in the IF output and that are sufficient to transform the annotated IF output in such a way that we define the channels (both in a cryptographic and an ideal model) in a meaningful way. Before we discuss these IF transformations, it is thus necessary to first summarize the main characteristics of IF and its extensions we consider here.

2. We introduce here and elsewhere "new" keywords and symbols, i.e. ones for which we assume that they do not occur in the given AnB• specification.

3. If there is no such step than the specification is rejected as *ill-goaled*.

Channel type	AnB•	AnB (+ annotations) for channels as assumptions	AnB (+ annotations) for channels as goals (sender/receiver side)
Authentic	$A \bullet\!\!\rightarrow B : M$	$A \rightarrow B : \mathsf{aAnnA}, A, B, M$	$\mathsf{aAnnGS}, A, B, h(M)$ and $\mathsf{aAnnGR}, A, B, h(M)$
Confidential	$A \rightarrow\!\!\bullet B : M$	$A \rightarrow B : \mathsf{cAnnA}, B, M$	$\mathsf{cAnnGS}, A, B, h(M)$ and $\mathsf{cAnnGR}, A, B, h(M)$
Secure	$A \bullet\!\!\rightarrow\!\!\bullet B : M$	$A \rightarrow B : \mathsf{sAnnA}, A, B, M$	$\mathsf{sAnnGS}, A, B, h(M)$ and $\mathsf{sAnnGR}, A, B, h(M)$

Table 1. Transformation between AnB• and AnB specifications for standard channels as assumptions and goals.

3. The Intermediate Format IF

The AVISPA Intermediate Format IF is a more low-level (with respect to AnB) language that is designed to specify transition systems in a way that is feasible for protocol analysis tools. As we remarked above, it is convenient to define the semantics of AnB by translation to IF (see [7], [31] and similarly [17], [24], [27], [29]) which has itself a clear and well-defined semantics. While AnB has a quite limited expressiveness (e.g. we cannot express protocols with repeated parts), IF does not have such limitations: we can specify the evolution of arbitrary concurrent processes.

In this paper, we use IF as part of a formal framework in which we define and relate channels. In particular, we introduce new symbols in IF for the different channel types when used as assumptions and as goals. A priori, these new symbols are not interpreted in a special way and we define their precise meaning in the following.

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 specifies which states count as attack states. As expected, a protocol is called *safe* when no attack state is reachable from the initial state using the transition relation.

An initial state is a finite set of ground terms. More generally, a state is a set of *facts* such as $\mathsf{iknows}(m)$, which expresses that in this state the intruder knows message m. This kind of fact has the particular property that it is *persistent*: when it is present in a state, then it is also present in all successor states, because the intruder never forgets messages. This simplifies the formulation of rules below.

Another central kind of fact is $\mathsf{state}_{\mathcal{A}}(A, m_1, \ldots, m_n)$, which characterizes the local state of an honest agent during the protocol execution by the messages $A, m_1, \ldots, m_n$ (the local knowledge of that agent). The constant $\mathcal{A}$ identifies the role of that agent, and, by convention, the first message A is the name of the agent.[4] We will later introduce further kinds of facts.

The *rules* of an IF specification are of the form

$$l \Rightarrow r$$

for sets of facts l and r, where the variables of r are a subset of the variables of l. For a *ground* state S (that has no variables), the rule defines transitions to the ground states $\{(S \setminus l\sigma) \cup r\sigma \mid l\sigma \subseteq S\}$, where σ is a substitution of the rule variables that *matches* the left-hand side l with a subset of S, and the successor states are obtained by removing the matched facts from the state and introducing the respective right-hand side facts under σ.

For instance, we can define the intruder model (in the style of Dolev-Yao) by a set of rules as follows:

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

The persistence of $\mathsf{iknows}(\cdot)$ allows us to omit the left-hand side $\mathsf{iknows}(\cdot)$ facts on the right-hand side (they are not removed). The first rule describes both asymmetric encryption and signing. The second rule expresses that an encrypted message can be decrypted when knowing 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).

We may have several further rules of the above form for intruder deduction. As it 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 following rule:

$$\mathsf{iknows}(T_1). \cdots .\mathsf{iknows}(T_n) \Rightarrow \mathsf{iknows}(f(T_1, \ldots, T_n)) \ .$$

We assume that all constants that represent agent names and public keys are public symbols (of arity 0).[5] In addition to this, we may also consider algebraic properties such as $\exp(\exp(g, X), Y) \approx \exp(g, Y), X)$ that we need for Diffie-Hellman key-exchanges. Moreover, one may model decryption by algebraic properties such as $\{\{M\}_K\}_{\mathsf{inv}(K)} \approx M$. While we allow for algebraic properties in general, for the results we are interested in here we assume that the symbols $\{\cdot\}.$ and $\langle \cdot, \cdot \rangle$ that we use in our model are interpreted in the free algebra. In § 4.1, we will extend this intruder model with rules for channels.

There are several extensions to this rule notion: first, a rule may contain the creation of fresh values, denoted as

$$l =\!\![V]\!\!\Rightarrow r$$

4. In contrast, the AnB notation does not distinguish between a role name such as $\mathcal{A}$ and the protocol variable A that is instantiated in the protocol execution with a concrete agent name like a playing role $\mathcal{A}$.

5. This assumption is also possible in case of an infinite set of agents, e.g. using an initialization theory like the one in [30].

for a list of variables V. The semantics of this extension is that the variables are replaced for some fresh constants during rule application. Further, one may write

$$l \mid cond \Rightarrow\!\![V]\!\!\Rightarrow r$$

to denote the addition of negative conditions and facts to the left-hand side of a rule, namely conditions of the form $\mathsf{not}(f)$ and $s \neq t$ for a fact f and terms s and t; but note that $\mathsf{iknows}(\cdot)$ facts may not occur negatively. The semantics is that this excludes from the transition all the substitutions under which f is contained in the current state (for a substitution of the remaining variables) and under which s and t are equal.

The transition rules of honest agents usually have the form

$$\begin{aligned}&\mathsf{state}_{\mathcal{R}}(msglist).\mathsf{iknows}(msg_{in})\\ &\qquad =\![V]\!\!\Rightarrow \mathsf{iknows}(msg_{out}).\mathsf{state}_{\mathcal{R}}(msglist')\ .\end{aligned}$$

This rule describes the transitions of an honest agent who plays the role $\mathcal{R}$ of the protocol and whose current local state is described by the list of messages $msglist$. This agent can receive a message msg_{in}, create a list of fresh values V, update its local state to $msglist'$ and send the msg_{out} as a reply. Here, both the incoming and the outgoing message are on a completely insecure channel: the message that the agent receives is chosen by the intruder, and the message that it sends is received by the intruder (see also [30]).

For instance, the second transition of A for our example of Figure 2 looks as follows when using insecure channels:

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

The attack states are described in IF by a set of *attack rules* of the form

$$l \mid cond \Rightarrow l.\mathsf{attack}$$

for a special fact symbol attack. (For simplicity, we will just write the left-hand side for such attack rules.) Thus, these rules describe a set of states to which the attack rule matches, adding the attack fact. We can then define attack states simply as those that contain the attack fact.

4. Standard Channels as Assumptions

We now define two formal models for channels as assumptions: the ideal channel model, ICM, which describes the functionality of a channel in an ideal way, and the cryptographic model, CCM, which employs cryptography to achieve the same properties on the basis of insecure channels. We also show that the CCM implements the ICM in a certain sense. As summarized in Table 2, the basic idea is the following. For insecure channels, the incoming and outgoing messages in the rules of honest agents are represented by $\mathsf{iknows}(M)$ on the left-hand side and right-hand side, respectively. In the ICM, we replace these $\mathsf{iknows}(\cdot)$ facts by new fact symbols that represent messages sent on the particular type of channel — and we describe rules about the intruder sending and receiving such messages below. In the CCM, in contrast, we ensure the channel properties by encrypting and signing messages.

It is straightforward to extend the AnB language with the concept of channels as assumptions for the ICM and CCM: the translation from AnB to IF is the same except that incoming and outgoing messages on the authentic, confidential, and secure channels in the rules of honest agents have to be appropriately wrapped by channel facts or encryptions as in Table 2. The formal definition extending an arbitrary AnB compiler with channels is covered in § 4.2 and § 4.4.

4.1. 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 channel. We refer to these three facts with the term *ICM facts* or *channel facts*. Note that 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, because this information is relevant for their definition. By the IF requirement that the right-hand side variables must be a subset of the left-hand side and fresh variables, at least the sender must know the name of the receiver when sending the message (and its own name, of course). 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 to be persistent (i.e. every such fact that holds in some state also holds in all successor states). 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) like the channel variant that we discuss in § 8. Moreover, like for $\mathsf{iknows}(\cdot)$ facts, we do forbid the channel facts to occur negatively in rules.

Finally, we require that the channel facts do not occur in the initial state or the goals. For instance, the second transition of A for our example of Figure 2 looks as follows (cf. (1)):

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

A thus processes the incoming message only if the sender and receiver names of the authentic channel fact agree with the variables in the state fact of A. Note that, due to

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

Table 2. Translation from IF with annotations to the ICM and the CCM for channels as assumptions.

persistence, the left-hand side fact $\mathsf{athCh}_{B,A}(GY)$ is not removed by applying this rule.

Note also that 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.

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, 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. For now, we do not further consider this, and simply assume that $\mathsf{dishonest}(A)$ holds for any dishonest agent name A in the initial state (i.e. only for $A = i$ in the classical intruder model). In general, we can allow IF rules that model the compromise of an agent A or the creation of a new dishonest identity A, where we have the predicate $\mathsf{dishonest}(A)$ on the right-hand side. However, our main motivation for the generalization is that we consider pseudonyms in § 7 including the possibility to arbitrarily generate new pseudonyms; whenever the intruder generates a pseudonym P this immediately induces the fact $\mathsf{dishonest}(P)$.

We conclude the definition of the ICM with its most essential part, namely the rules for the intruder sending and receiving on the channels. The rules in Table 3 define the abilities of the intruder on these channels and thus their ideal functionality:

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

Table 3. The intruder rules for the ICM.

(3) He can send messages on an authentic channel only under the name of a dishonest agent A to any agent B.[6]
(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.

4.2. Translating AnB• to ICM

We now define how to transform an IF specification with annotations into a specification in the Ideal Channel Model ICM, which describes the functionality of a channel in an ideal way. This transformation completes the translation process of an AnB• specification into an ICM specification (cf. Figure 1).

For this transformation, we transform the rules by replacing $\mathsf{iknows}(m)$ facts that carry any of the aAnnA, cAnnA, or sAnnA annotations, i.e. that match an entry in the second column of Table 2, with the fact for the corresponding channel type for the incoming or outgoing message, i.e. as in the third column of the table.

Consider again the second transition of A for our example of Figure 2 (cf. (1) and (2)). The annotated IF rule that comes out of the standard AnB compiler for A receiving this

6. The intruder knows all agent names by assumption, but we need $\mathsf{iknows}(B)$ on the left-hand side because IF requires all variables that appear on the right-hand side to be already present on the left. (Similarly for other rules.)

message looks as follows (where the next outgoing message of A is on an insecure channel, as in Figure 2 (cf. (1)):

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

This rule is translated in the ICM transformation into the rule (2).

4.3. 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 correct (as we show below) cryptographic realization.

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 with the term *CCM material*. We assume that every agent, including the intruder, knows initially both keytables and its own private keys. Thus the additional initial intruder knowledge of the crypto channel model is

$$\{\mathsf{ak},\mathsf{ck},\mathsf{inv}(\mathsf{ak}(i)),\mathsf{inv}(\mathsf{ck}(i)),\mathsf{atag},\mathsf{ctag},\mathsf{stag}\}\,. \qquad (9)$$

For instance, the second transition of A for our example of Figure 2 looks as follows (cf. (1) and (2) in the ICM):

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

A thus processes the incoming message only if the sender and receiver names of the authentic channel fact agree with the variables in the state fact of A.

The encoding we have chosen is sufficient for our purposes, but there are several possible alternatives for such an implementation. For instance, one may choose not to distinguish between signing and encryption keys (and thus identify ak with ck), but has to ensure then that signing never accidentally decrypts an encrypted message.

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: $\{\{\mathsf{stag},M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$. A dishonest B 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. [14]. 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.

The tags are public constants that determine the meaning of a message (from the point of view of the agent that generated the message), and the receiver will only accept messages that have the correct tag according to the protocol. The encryption should ensure that:

- only the agent that can generate the message in the CCM can then send it on the respective channel in the ICM, and similarly, that
- only the agent that can receive the message in the CCM can then receive it on the respective channel in the ICM.

In fact, this will be shown in the following section.

4.4. Translating AnB• to CCM

We transform the rules by replacing messages that carry any of the aAnnA, cAnnA, or sAnnA annotations, i.e. that match an entry in the second column of Table 2, with the corresponding encoding in the rightmost column.[7]

For instance, the example rule $(*)$ is translated in the CCM transformation into the following rule:

$$\begin{array}{r}\mathsf{state}_{\mathcal{B}}(B,\ldots).\mathsf{iknows}(\{\mathsf{atag},B,M\}_{\mathsf{inv}(\mathsf{ak}(A))})\\ =[V]\Rightarrow\ \mathsf{iknows}(M').\mathsf{state}_{\mathcal{B}}(B,\ldots)\end{array}$$

4.5. 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 CCM.

A simulation in the other direction does not directly work (since the intruder can use the cryptographic tables to produce messages that cannot appear in the ICM), but requires 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.

7. Alternatively, we could obtain a CCM specification from an ICM one, e.g. by replacing all transition rules of honest agents in the ideal channel model specification with encrypted messages on the insecure (intruder-controlled) channel according to Table 1.

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 specification, 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, and 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 (e.g. $\mathsf{athCh}_{A,B}(M)$) that occurs in the rules of R_1 with the corresponding intruder knowledge of the CCM (e.g. $\mathsf{iknows}(\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))})$).*

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 say that $S_1 \sim S_2$ 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 2.*

In the $\sim$ relation, we do not consider an intruder knowledge that is closed under intruder deduction, but rather only the $\mathsf{iknows}(\cdot)$ facts. We then have the following two theorems; actually, since Theorem 1 and Theorem 2 are special cases of the more general theorems for pseudonyms (Theorem 4 and Theorem 5), we state these first results here and prove only the more general results in the following.

Theorem 1. *Consider an ICM specification and the corresponding CCM specification. For a reachable state S_1 of the ICM specification, there is a reachable state S_2 of the CCM specification such that $S_1 \sim S_2$.*

To establish the converse direction, we need two additional assumptions. 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 [23], 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 $\{\!|\cdot|\!\}.$ 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*.

Note that these assumptions are sufficient for the following theorem but not necessary.

Theorem 2. *Consider an ICM specification and the corresponding CCM specification, 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 $S_1 \sim S_2$.*

These two theorems together formally 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.[8]

5. Standard 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. [7], [26], [30]. We first discuss this for IF in general and then for the translation from AnB•to IF.

8. For example, constraint-based approaches like OFMC [9] do not have a problem with complex message terms (as present in the CCM), but rather with models that allow for many interleavings of the actions (which is the case for ICM). For SATMC [6], for instance, the situation is just the opposite. Also, the CCM requires only cryptographic features that all protocol analysis tools offer, so it can be used without extending tools.

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. 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

$$A \ channel \ B : M$$

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 the message 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.[9]

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 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.

9. 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.

We can now define attacks in a protocol-independent way based on the attack states in Table 4. The rules (10) and (12) reflect the standard definition of secrecy and authentication goals (non-injective agreement in the terminology of [26]; the injective variant is considered in § 8). 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. 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 i to arbitrary identities controlled by the intruder (in accordance to what we said about the intruder model in § 4.1).

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 composability result in § 6. 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

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

with the goal $A \bullet\!\!\rightarrow B : M$. The intruder 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. Intuitively, 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.

5.1. Translating AnB• goals to IF

We use the annotations for the goals that we have introduced in the transformation from AnB• to AnB and that the IF is annotated with. We perform the transformation summarized in Table 5 on the IF file. We must distinguish here between sending and receiving messages, i.e. right-hand side and left-hand side, respectively, $\mathsf{iknows}(\cdot)$ facts of the IF transition rules. We look first for occurrences of the annotation $\mathsf{aAnnGS}, A, B, h(M)$. If it occurs on a left-hand side of a rule, we simply remove it (because the message is not sent in this rule). If it occurs on the right hand side, we remove it from the message and add the

$$\mathsf{request}(A,B,P,M) \quad | \quad \mathsf{not}(\mathsf{witness}(A,B,P,M)), \mathsf{not}(\mathsf{dishonest}(A)) \tag{10}$$
$$\mathsf{request}(A,B,P,M).\mathsf{dishonest}(A) \quad | \quad \mathsf{not}(\mathsf{iknows}(M)) \tag{11}$$
$$\mathsf{whisper}(B,P,M).\mathsf{iknows}(M) \quad | \quad \mathsf{not}(\mathsf{dishonest}(B)) \tag{12}$$
$$\mathsf{hear}(B,P,M) \quad | \quad \mathsf{not}(\mathsf{whisper}(B,P,M)), \mathsf{not}(\mathsf{iknows}(M)) \tag{13}$$

Table 4. Attack states for defining channels as goals.

Channel	Annotated IF	Goal Facts
Authentic/Sending	aAnnGS, A, B, M	$\mathsf{witness}(A,B,P,M)$
Confidential/Sending	cAnnGS, B, M	$\mathsf{whisper}(B,P,M)$
Secure/Sending	sAnnGS, A, B, M	$\mathsf{witness}(A,B,P,M).\mathsf{whisper}(B,P,M)$
Authentic/Receiving	aAnnGR, A, B, M	$\mathsf{request}(A,B,P,M)$
Confidential/Receiving	cAnnGR, B, M	$\mathsf{hear}(B,P,M)$
Secure/Receiving	sAnnGR, A, B, M	$\mathsf{request}(A,B,P,M).\mathsf{hear}(B,P,M)$

Table 5. Second part of the IF transformation for channels as goals.

fact $\mathsf{witness}(A,B,P,h(M))$ to the right-hand side, where P is a unique identifier of the protocol (which we need for the composability later). We handle occurrences of the annotations cAnnGS and sAnnGS in a similar way.

For what concerns aAnnGR, A, B, X, we simply remove it when it occurs on a right-hand side (since here it is not received). If it occurs on the left-hand side, we first check that $X = h(M)$. If this is not the case, then the receiver cannot see the message M at this point from the received message and thus not check the hash value here. In this case we have simply a specification mistake: the channel implementation is broken such that the receiver cannot receive the message. Otherwise, if $X = h(M)$, we remove the term and add to the right-hand side the fact $\mathsf{request}(A,B,P,M)$, where again P is the identifier of the protocol. We proceed similarly for occurrences of the tags cAnnGR and sAnnGR.

Using our annotations, we are thus able to generate variants of the rules where special predicates are introduced for sending and receiving the messages whose authentic, confidential, or secure transmission was declared as a goal of the protocol.

6. Compositional Reasoning for Standard 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, e.g. using disjoint key-spaces for the protocol. We do not want to commit to particular such composition arguments nor to dive into the complex argumentations behind this. For this reason, we 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, 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. One may thus wonder whether this definition is useful at all. To see that it 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 with $P_2[P_1]$ 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). 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 two 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.*[10] *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.*

10. 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.

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 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 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 $P_2[P_1]$.*

This theorem is a special case, and thus directly follows from, the more general theorem for pseudonyms (Theorem 6) that we prove in the appendix.

Example 1. *Consider our example protocol of Figure 2 as P_2 and let us implement the first authentic channel by the following protocol P_1:*

$$\begin{array}{cccl} A' & \rightarrow & B' & : \ \{B', M'\}_{\mathsf{inv}(\mathsf{pk}(A'))} \\ \hline A' & \bullet\!\!\rightarrow & B' & : \ M' \end{array}$$

The composition $P_2[P_1]$ is then:

$$\begin{array}{cccl} A & \rightarrow & B & : \ \{B, \exp(g, X)\}_{\mathsf{inv}(\mathsf{pk}(A))} \\ B & \bullet\!\!\rightarrow & A & : \ \exp(g, Y) \\ A & \rightarrow & B & : \ \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : \ Payload \end{array}$$

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^*.

Now consider the following variant P_2' of the Diffie Hellman key-exchange P_2 that we intentionally designed so *that it breaks when composing it with P_1:*

$$\begin{array}{cccll} A & \rightarrow & B & : & \{B, g\}_{\mathsf{inv}(\mathsf{pk}(A))} \\ A & \bullet\!\!\rightarrow & B & : & \exp(g, X) \\ B & \bullet\!\!\rightarrow & A & : & \exp(g, Y) \\ A & \rightarrow & B & : & \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : & Payload \end{array}$$

In the additional first message, A transmits (authentically) the basis g 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'

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

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.*

7. Pseudonymous Channels

Pseudonymous channels are created by techniques like purpose-built keys (PBK) or TLS without client authentication: we have something similar to a secure channel except that one end is not identified by its real name but by some pseudonym, which is usually related to an unauthenticated public-key; see, e.g., [11], [18], [22], [25]. 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 real identity is not known or not guaranteed.[11] However, there is more to it.

First, pseudonymous channels, both as assumptions and as goals, should not be defined as entirely new concepts unrelated to the previous channels. Rather, we define them as variants of the standard channels discussed above where one (or both) ends are identified by a pseudonym rather than the real name.[12]

Second, the concept of pseudonymous secure channels is useful to model a number of scenarios. The most common one is probably the above mentioned TLS without client authentication as it is common in the Internet: it is in a sense weaker than a standard secure channel, but (assuming the server's public key is properly authenticated) it is sufficient for submitting a client's password over this channel to achieve full authentication. We thus want to use such a channel both as goal for protocols like TLS where only one side is authenticated, and as an assumption in high-level protocols that use such a channel for a login, for instance.

Another, quite different example is a smart card in a card reader: here we have communication between two entities that are not yet authenticated (and will engage in a protocol to achieve authentication) but that have a secure pseudonymous channel in the sense that an attacker cannot interfere with their communication. In this case, the physical environment gives us the secure pseudonymous channel that authentication protocols may use as an assumption.

7.1. AnB• notation for pseudonymous channels

We extend the AnB• notation for standard channels and 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). The omitted variant is a short-cut for a pseudonym that A freshly generates when it first uses a pseudonymous channel. We extend AnB here with a new type symbol for the type declarations, namely the type "pseudonym", which may be used in place of agent names.

We use a similar notation for the other kinds of pseudonymous channels.[13] For a pseudonymous secure channel where both ends can be pseudonymous, we generally exclude that sender and receiver are identical pseudonyms (i.e. honest agents check that they never send messages to themselves).

Example 2. *Consider the following protocol that establishes a secure channel between an unauthenticated A, which uses its Diffie-Hellman half-key* $\exp(g, X)$ *as a pseudonym, and an authenticated B (just as in the case of TLS).*

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

Such a channel is good enough for a login protocol, e.g.

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

where $Payload'$ is now on a standard secure channel (assuming that the password of A is sufficient to authenticate her to the server B).

11. [22] first formalized the notion of sender invariance as a goal, which is similar to our notion of a pseudonymous-authentic channel. However, in [22], it is defined directly as messages coming from the same source, while we see it here simply as a classical authentication goal with a pseudonymous sender.

12. One may even argue that real names are also just a kind of pseudonym, so there is no difference at all. In our model, the difference between real names and pseudonyms is that we assume that real names uniquely identify agents and do not change over time, while pseudonyms may be arbitrarily created by any agent. As a consequence, every agent (including the intruder) can act under several identities.

13. Pseudonymization on insecure end-points such as $A \bullet\!\!\rightarrow [B]$ is not useful and we thus allow the $[\cdot]$ only on the bulleted end-points of channels.

We assume, for executability of the specification, that as soon as a new pseudonym is created, it is transmitted to the other agents. Moreover, it is crucial that pseudonyms cannot be "stolen", i.e. they belong to the agent who created them. We will see below how this is actually implemented when we consider the cryptographic model of channels, but we first give an ideal model that is based on defining pseudonymous channels in analogy to standard ideal channels.

7.2. The ICM for pseudonymous channels

The transformation on the AnB• specifications are as before, and we extend the ICM with pseudonymous channels using IF facts as before. The difference is that we use identifiers that may both be real names and pseudonyms. For instance, receiving a message from an agent acting under pseudonym ψ on an authentic channel looks as follows:

$$\begin{array}{l}\mathsf{state}_{\mathcal{B}}(\ldots).\\ \mathsf{athCh}_{\psi,B}(M)\\ =[V]\Rightarrow\\ \ldots\end{array}$$

We must distinguish two cases here. One is that B does not know the pseudonym ψ in advance, i.e. it is not part of his state fact. This holds for the first transmission from ψ in the current run. The other case is given by the other transitions where B already has the pseudonym in his state fact and does not accept messages from a different sender with a new pseudonym. This is exactly the notion of sender invariance, of course.

The rules for the intruder remain the same except that we must ensure that the intruder can generate himself new pseudonyms at any time and can send and receive messages with these new pseudonyms. To keep track of this, we use again the predicate $\mathsf{dishonest}(\cdot)$. Hence, the rule generating a new pseudonym ψ in the ICM and the CCM now is:

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

Here, the knowledge of $\mathsf{inv}(\psi)$ is only needed later for the CCM model where the pseudonyms are all in fact public keys — the intruder must know all the respective private keys of dishonest pseudonyms. Using the predicate $\mathsf{dishonest}(\cdot)$ allows us to work with exactly the same channel rules as before, only that instead of agent names, we may now also use pseudonyms as sender or receiver names.[14]

7.3. The CCM for pseudonymous channels

A straightforward way to realize pseudonymous channels is based on asymmetric key-pairs built for this purpose, where the public key (or a hash value thereof) is used as the pseudonym, e.g. [11]. We define the ownership of such a pseudonym by the knowledge of the corresponding private key. As usual, we assume that private keys of honest agents are never leaked and the intruder cannot obtain any private keys except those he created himself. Under these reasonable assumptions, the intruder is not able to "steal" the pseudonyms of honest agents. Rather, every agent remains the single owner of all pseudonyms it created. Note that the intruder can arbitrarily generate pseudonyms of his own.

To connect this notion of pseudonyms with channels, we need to ensure that only the owner of a pseudonym can access a secured pseudonymous endpoint of a channel with that pseudonym. In fact, we can use the same encoding as in the CCM before, with the only difference that when the sender or receiver name is a pseudonym, then it is directly the public key. That is, we do not use $\mathsf{ak}(A)$ or $\mathsf{ck}(A)$ as before, but we directly use the pseudonym ψ for encryption or the corresponding private key $\mathsf{inv}(\psi)$ for signing. (Note that we have excluded the case that sender and receiver are the identical pseudonym, before.) For instance, an authentic channel from a sender with pseudonym ψ to an agent B, where B is either an agent name or a pseudonym, is then encoded by $\mathsf{iknows}(\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\psi)})$. The transformations on the AnB• specification are as before, only we need to keep track of whether we are dealing with an agent name or a pseudonym, because the translation for CCM depends on this.

7.4. Relating the two models

After generalization of the ICM and CCM models with pseudonymous channels, we still have the property that every attack against the ICM can be simulated in the CCM:

Theorem 4. *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 $S_1 \sim S_2$.*

Vice-versa, under the assumptions of typing and full receiver decryption, all attacks in the CCM can be simulated in the ICM:

Theorem 5. *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 $S_1 \sim S_2$.*

7.5. Pseudonymous channels as goals

We use again the idea that pseudonymous channels are like standard channels with the only exception that the

14. In a typed model, however, this gives a conflict because of the sender or receiver types. We may either redefine these predicates to allow for a larger set of identifiers (agent names and pseudonyms), or define new predicates (or overload them) with respective different types and exactly the same rules except for the types.

secured endpoints are logically tied to pseudonyms instead of real names. Therefore, we define these goals as variants of the standard secrecy and authentication goals with pseudonyms instead of real names.

Recall that in the case of pseudonymous channels as assumptions, when no pseudonym is specified, the respective agent freshly creates one at the beginning of a session. This is no longer possible when using pseudonymous channels as goals, because the pseudonym would then be unrelated to the protocol messages and thus the goal would trivially be violated. In fact, whatever the pseudonym of an agent in a pseudonymous channel goal is, it needs to be somehow related to the protocol messages. We thus require that for pseudonymous channels as goals, we always explicitly annotate what the pseudonym is.

The most common example would be a protocol where A has an unauthenticated public-key $\mathsf{pk}(A)$ and the protocol ties transmissions to this public-key as in the CCM above. Then, a goal declaration may be for example:

$$[A]_{\mathsf{pk}(A)} \bullet\!\!\rightarrow\!\!\bullet B : M \ .$$

There is a requirement for what can be used as a pseudonym: the owner of a pseudonym P is honest iff $\mathit{honest}(P)$ holds. In the above example, thus $\mathit{honest}(\mathsf{pk}(A))$ must hold iff $\mathit{honest}(A)$ holds. Besides that, our definition does not impose any further requirements on the term that can be used as a pseudonym—this is a simply matter of specification: it counts as an attack if the protocol does actually not ensure the transmission properties with respect to the declared pseudonym. For instance, in the above example, it would count as an attack if B receives a message M that apparently comes from $\mathsf{pk}(A)$ for an honest A, but A has never said M to B.

While it is straightforward that the public key of an agent can be used as pseudonym when messages are indeed secured by it, the cases where the pseudonymous agent uses no public keys are not obvious. We discuss in the following two such cases that often occur in practice.

7.5.1. Unauthenticated Diffie-Hellman Key-Exchange.

Common examples are protocols with a Diffie-Hellman key-exchange where one side is not authenticated as in the following example:

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

Here, we use the half-key $\exp(g,X)$ as a pseudonym for A: in fact one may consider X as a private key and $\exp(g,X)$ as the corresponding public-key.[15] Intuitively, the reason why this works is that, from the point of view of an honest agent B, $\mathit{Payload}$ can only originate from the owner of the pseudonym $\exp(g,X)$, i.e. the one who created X. But, as before, formally the point is merely that the protocol goal holds with respect to this pseudonym.

7.5.2. TLS.

Consider now the case of TLS without client authentication. Indeed, one may simply deploy the protocol exactly as in the authenticated case, except that the key-certificate for A is not signed by a trusted party but by A itself. (Note that this public-key may even be created freshly in each new session.) Then, we can use this public-key conveniently as the pseudonym for the goal, i.e. $[A]_{\mathsf{pk}(A)} \bullet\!\!\rightarrow\!\!\bullet B : M$ where M is a payload message from A to B.

The actual deployment of TLS without client authentication however does not bother about public keys of clients A; rather, the pre-master secret is simply encrypted for B (who is authenticated by a public-key certificate), but not signed by A. Let us illustrate this by the following, further simplified, protocol:

$$\begin{array}{rl} A \!\rightarrow\!\bullet B : & \mathit{PMS} \\ A \rightarrow B : & \{\!|\mathit{Payload}|\!\}_{\mathit{PMS}} \\ \hline [A]_{\mathit{PMS}} \bullet\!\!\rightarrow\!\!\bullet B : & \mathit{Payload} \end{array}$$

Obviously, there is now a problem with using PMS as a pseudonym. The reason is that the creator has no control over the usage of the pseudonym in contrast to the case of a public key with a corresponding private key that remains secret. This is illustrated by the following "attack" where a starts a session with the dishonest i:

$$\begin{array}{rl} a \!\rightarrow\!\bullet i : & \mathit{pms}_a \\ a \rightarrow i : & \{\!|\mathit{payload}_{17}|\!\}_{\mathit{pms}_a} \\ & \mathsf{witness}(\mathit{pms}_a, i, p, \mathit{payload}_{17}) \\ i \rightarrow b : & \mathit{pms}_a \\ i \rightarrow b : & \{\!|\mathit{payload}_{17}|\!\}_{\mathit{pms}_a} \\ & \mathsf{request}(\mathit{pms}_a, b, p, \mathit{payload}_{17}) \end{array}$$

So we have a request event, created by b, referring to the pseudonym pms_a of an honest agent. But actually, intuitively, there is nothing wrong with this trace: we could have a corresponding trace for a variant of the protocol where pms_a is signed, together with the intended recipient, by a private key corresponding to an unauthenticated public-key of a which is then the pseudonym. The crucial difference of this public-key variant is that the intruder cannot use the

15. One may wonder here whether this should indeed be called public-key cryptography, since X and $\exp(g,X)$ cannot be directly used for the encryption and decryption of messages, but are the bases for the derivation of shared keys. However, when defining public-key cryptography in a general sense as "a means to turn authentic into secure channels", Diffie-Hellman key-exchange is indeed a form of public-key cryptography (in the spirit of what Diffie and Hellman themselves observe in [19]).

same pseudonym that a has used before, but has to use one of his (dishonest) pseudonyms.[16]

So, in fact, we have the situation that the goal

$$[A] \bullet\!\!\rightarrow\!\!\bullet\, B : \mathit{Payload}$$

seems to hold *in some sense*, even though we cannot identify a pseudonym that the intruder could never steal. The idea is now that an honest A freshly creates the pseudonym PMS in each session; this creation is with respect to the agent B to whom A wants to talk to—and the intruder can "steal" the pseudonym only iff this B is dishonest, while pseudonyms that an honest A creates for an honest B are all safe. Thus, if we define for this protocol that $\mathit{honest}(PMS)$ holds iff it is created by an honest A for an honest B, then the intruder cannot steal honest pseudonyms and the goal actually holds with respect to this definition.

This construction must be reflected in the specification for the goal, and we thus write such a goal as follows:

$$[A]_{PMS^B} \bullet\!\!\rightarrow\!\!\bullet\, B : \mathit{Payload}$$

meaning that A acts under pseudonym PMS that is created for the recipient B.

Note that this case (secure channels with respect to an unauthenticated agent, identified by a shared key) appears also in many other protocols that can be handled similarly, e.g. Geopriv [32].

7.6. Compositional reasoning for pseudonymous channels

The definition of secure pseudonymous channels as assumptions and as goals differs from the real-name cases only by the use of pseudonyms (and the attached notion of honesty), but the rest of the construction is exactly as before. This allows us also to extend our compositionality result to pseudonymous channels:

Theorem 6. *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 $P_2[P_1]$.*

8. Towards Freshness

As a final contribution, we consider a further variant of channels, namely ones that ensure freshness of messages, i.e. suppress the replay of messages. It is obvious that the CCM allows such a replay if the payload messages do not contain a mechanism to prevent it, since the intruder can replay encrypted messages that he has seen any number of times even if he cannot decypt them. The ICM models this property, too.

$$\begin{aligned}
\mathsf{dishonest}(A).\mathsf{iknows}(B).\mathsf{iknows}(M) &\;=\!\![N]\!\!\Rightarrow\; \mathsf{fathCh}^{N}_{A,B}(M)\\
\mathsf{fathCh}^{N}_{A,B}(M) &\;\Rightarrow\; \mathsf{iknows}(M)\\
\mathsf{dishonest}(A).\mathsf{iknows}(B).\mathsf{iknows}(M) &\;=\!\![N]\!\!\Rightarrow\; \mathsf{fsecCh}^{N}_{A,B}(M)\\
\mathsf{fsecCh}^{N}_{A,B}(M).\mathsf{dishonest}(B) &\;\Rightarrow\; \mathsf{iknows}(M)
\end{aligned}$$

Table 6. Additional intruder rules for fresh channels.

We denote the variant of authentic and secure channels that prevent replay as follows (for confidential channels, the freshness is not an issue):

- *Fresh-authentic channel*: $A \bullet\!\!\twoheadrightarrow B : M$. This channel is like an authentic channel with the restriction, intuitively, that M can only be received once.
- *Fresh-secure channel*: $A \bullet\!\!\twoheadrightarrow\!\!\bullet\, B : M$. A channel that is both fresh-authentic and confidential.

The formal definition of these variants in the ICM and the CCM is similar to before, where we have to introduce new annotation symbols for channels as assumptions and goals and translation processes on AnB and IF.

8.1. Fresh channels as assumptions in the ICM

For the ICM, let $\mathsf{fathCh}^{N}_{A,B}(M)$ and $\mathsf{fsecCh}^{N}_{A,B}(M)$ be the facts used for fresh-authentic and fresh-secure channels as assumptions. In contrast to their non-fresh counter-parts $\mathsf{athCh}_{A,B}(M)$ and $\mathsf{secCh}_{A,B}(M)$, these facts are *not* persistent, because once the message is received by an honest agent (i.e. matched in the left-hand side of a transition rule of an honest agent) the fact is removed from the state and the respective message cannot be received anymore. This models exactly the replay protection. However, since a state is a set of facts, this non-persistence introduces the problem that an agent cannot send twice exactly the same message for the same recipient before that recipient actually receives it (as the state cannot contain multiple copies of the fact at the same time, and the only copy is "consumed" with the first receiving transition). To allow multiple sending of the same message anyway, we have introduced a further parameter N into these facts that contains simply a fresh number that is irrelevant for the receiver but that distinguishes multiple occurrences of otherwise identical facts.

For the intruder, we have now the rules for fresh-authentic and fresh-confidential channels shown in Table 6. These rules are similar to their non-fresh counter-parts with only the difference that we have the fresh nonce that the intruder has to also generate (so he can also send an arbitrary number of identical messages on the fresh channels). These

16. One may still argue that the intruder is able to "steal" the pseudonym pms_a created, and thus "owned", by a and authenticate with respect to that pseudonym. However, the protocol does not define PMS to be a pseudonym—that is a view we have just introduced.

definitions apply both for real names and for pseudonyms, of course.

8.2. Fresh channels as assumptions in the CCM

In the CCM, we use a simple mechanism to ensure the freshness of messages, namely a challenge-response mechanism. The idea can best be described as a reduction on the AnB•/AnB level, namely regarding the channel $A \bullet\!\!\twoheadrightarrow B : M$ (when used as an assumption) as syntactic sugar for the message exchange:

$$\begin{array}{lll} B \to A & : & N \\ A \bullet\!\!\to B & : & N, M \end{array}$$

where N is a new symbol representing a fresh challenge created by B (note that this N is different from the identifier N used in the ICM to disambiguate identical message transmissions).

The idea is thus that the receiver B creates a fresh (unpredictable) challenge N that must be contained in the received message. Moreover, for each time an honest agent A sends the message M, it can be received at most one time by an honest agent B, because on multiple receipts different challenges would have been created and thus included in the transmission.

A similar transformation is applied for reducing fresh-secure to secure channels. Observe that the intruder can read the message M, but only use one challenge-response from an arbitrary agent B. Again, we can extend this notation for the use of pseudonyms as expected.

8.3. Fresh channels as goals

Recall that the (non-fresh) authentic channels, when used as goals, are characterized by two properties (expressed as attack rules), namely the property (10) that corresponds to non-injective agreement [26] and the property (11) that the intruder actually knows all messages that come from any dishonest agent. For the fresh-authentic channels, we introduce an additional attack rule, (14), that defines replay of a message as an attack. (10) and (14) together correspond to Lowe's notion of injective agreement [26]:

$$\begin{array}{l} \mathsf{frequest}(A, B, P, M, N).\mathsf{frequest}(A, B, P, M, N') \\ \qquad\qquad \mid \mathsf{not}(\mathsf{dishonest}(A)) \wedge N \neq N' \end{array} \qquad (14)$$

We have here introduced a variant of the request fact, $\mathsf{frequest}(\cdot)$, that contains a unique identifier N for each occurrence (similar to the one of $\mathsf{fathCh}^{N}_{A,B}(M)$ in the ICM).

The assumption on which our construction is based is that the payload message M contains something fresh like a nonce (as we have also assumed in § 6). Under this assumption, a replay attack occurs whenever a message that supposedly originates from an honest agent A is successfully received by B more than once. This simple trick allows us to avoid counting $\mathsf{witness}(\cdot)$ and $\mathsf{request}(\cdot)$ events, which would be necessary if exactly the same message could be sent several times.

We are currently working at extending theorems 1–6 for the fresh-authentic and fresh-secure channels.

9. Related Work and Conclusions

We have given formal definitions of secure pseudonymous channels and proved the relationship between a model representing the ideal functionality of a channel and a model that employs concrete cryptographic messages on insecure channels. We have also shown a compositionality result that formalizes the replacement of an assumed channel with an arbitrary protocol that provides such a channel. We conclude by discussing relevant related works and pointing to directions for future research, in addition to those that we already mentioned above.

In [28], Maurer and Schmid introduce the • notation for authentic, confidential, and secure channels. They do not give formal definition of their channels and 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 have introduced in § 8. They use the notation to introduce a calculus for reasoning about what new 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).

Boyd [10] similarly reasons about what kind of relations between participants can be constructed, and what not, given a particular infrastructure of keys. His approach uses a formal model in Z, but it does not explicitly deal with protocols to achieve this.

Dilloway and Lowe [21] consider the specification of secure channels, used as assumptions, in a formal/black-box cryptographic model. They define several channel types, some of which are very similar to our standard authentic (namely their channel type $NF \wedge NH$), confidential ($C \wedge NR$), and secure channels ($C \wedge NF \wedge NR \wedge NH$). They do not have an equivalent for our strongest channels, fresh-authentic and fresh-secure channels as they do not consider the prevention of replay. They do, however, further differentiate properties of our channels, namely with respect to authentic channels they distinguish between hijacking (i.e. changing the apparent sender of a message) and faking (the intruder making up a message), and for confidential channels between confidentiality (the intruder learning the message) and redirecting (the intruder altering the destination of the message). Also, for redirecting and hijacking, they have two variants each, namely one where the intruder can perform changes with respect to either only honest agent names or for all agent names. This differentiation of channel properties can indeed make sense as they give implementations with cryptography (like we do) that have this property. However, we do consider less variants for the following reason. If B knows the public key for signatures of A, for instance, then B can verify signed messages from A; if A additionally includes the name of B in the message, the intruder cannot forward the signature to somebody else. Including this name means only a little extra cost, but improves the channel quality in the hierarchy of [21]. Thus, from the keys that some channels require as a realization, we can sometimes obtain a better channel at only little extra cost. Therefore, we do not consider the weaker channel in this case.

Like [21], Armando et al. in [4], [5] 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.

[22] first formalized the notion of sender invariance as a goal, which is similar to our notion of a pseudonymous-authentic channel. However, in [22], it is defined directly as messages coming from the same source, while we see it here simply as a classical authentication goal with a pseudonymous sender. Other notions that may be called receiver invariance, for instance, naturally result from the generalization of other channel types to pseudonymous agents. As we have discussed in § 7, these channel types are practically relevant, for instance when using protocols like TLS.

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. A similar approach is considered in [12].

Much effort has been devoted to the composition of protocols, e.g. [2], [15], [16] to name a few works. The closest to our work is [15]. In a formal setting similar to ours, they consider the sequential and parallel composition of protocols. In Theorem 6, which proves that we can replace an ideal channel with any protocol that implements it, we assume that the protocols in question are securely composable. We are currently investigating whether the approach of [15] is sufficient to achieve this assumption; a closer investigation is left for future work.

There are two frameworks for the secure composition of cryptographic primitives and protocols, namely the *Universal Composability* [13] Framework and the *Reactive Simulatability* [8] Framework. 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 with a non-negligible probability. The view of cryptography through indistinguishability from an ideal system is not directly feasible for the automated verification of security protocols.

We conclude with a final remark on the fact that all the arguments in this paper are within a black-box cryptography world and have not been related to cryptographic soundness. The transition from a cryptographic model to a black-box model in general implies the exclusion of (realistic) attacks; although for many application such models can be indeed shown to be cryptographically sound [33]. The simulation proofs of our theorems now tell us that we do not loose *further* attacks by going from the CCM to the ICM or vice-versa, or by verifying channel protocols and application protocols in isolation. We can thus can safely make the automated verification task easier. Besides this, the simulation also gives us insights in the properties of our formal models (e.g. relating ICM and CCM) 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 for valuable discussions and comments.

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 *Proceedings of CAV'05*, 2005.

[4] A. Armando, R. Carbone, and L. Compagna. LTL Model Checking for Security Protocols. In *Proceedings of CSF20*. IEEE Computer Society Press, 2007.

[5] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra Abad. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *Proceedings of FMSE 2008*. ACM Press, 2008.

[6] A. Armando and L. Compagna. SAT-based Model-Checking for Security Protocols Analysis. *International Journal of Information Security*, 6(1):3–32, 2007.

[7] AVISPA. Deliverable 2.3: The Intermediate Format. Available at www.avispa-project.org, 2003.

[8] M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems, 2004. Cryptology ePrint Archive, Report 2004/082, http://eprint.iacr.org/.

[9] D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[10] C. Boyd. Security architectures using formal methods. *IEEE Journal on Selected Areas in Communications*, 11(5):694–701, 1993.

[11] S. Bradner, A. Mankin, and J. Schiller. A framework for purpose built keys (PBK), June 2003. Work in Progress (Internet Draft: draft-bradner-pbk-frame-06.txt).

[12] M. Bugliesi and R. Focardi. Language based secure communication. In *Proceedings of CSF 21*, pages 3–16. IEEE Computer Society Press, 2008.

[13] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.

[14] I. Cervesato, A. D. Jaggard, A. Scedrov, J.-K. Tsay, and C. Walstad. Breaking and fixing public-key Kerberos. *Information and Computation*, 206(2–4):402–424, 2008.

[15] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, pages 11 – 23. ACM, 2003.

[16] S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *Proceedings of CSF 21*, pages 239–251. IEEE Computer Society Press, 2008.

[17] G. Denker, J. Millen, and H. Rueß. The CAPSL Integrated Protocol Environment. Technical Report SRI-CSL-2000-02, SRI International, Menlo Park, CA, 2000.

[18] T. Dierks and C. Allen. RFC2246 – The TLS Protocol Version 1, Jan. 1999.

[19] W. Diffie and M. Helmann. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6):644–654, 1976.

[20] C. Dilloway. Chaining secure channels. In *Proceedings of FCS-ARSPA-WITS'08*, 2008.

[21] C. Dilloway and G. Lowe. On the specification of secure channels. In *Proceedings of WITS '07*, 2007.

[22] P. Hankes Drielsma, S. Mödersheim, L. Viganò, and D. Basin. Formalizing and analyzing sender invariance. In *Proceedings of FAST'06*, LNCS 4691. Springer-Verlag, 2007.

[23] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proceedings of CSFW'00*. IEEE Computer Society Press, 2000.

[24] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and Verifying Security Protocols. In *Proceedings of LPAR 2000*, LNCS 1955, pages 131–160, 2000.

[25] D. Johnson, C. Perkins, and J. Arkko. RFC3775 – Mobility Support in IPv6, June 2004.

[26] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of CSFW 10*, pages 31–43. IEEE Computer Society Press, 1997.

[27] G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. *J. Comp.Sec.*, 6(1):53–84, 1998.

[28] U. M. Maurer and P. E. Schmid. A calculus for security bootstrapping in distributed systems. *J. Comp.Sec.*, 4(1):55–80, 1996.

[29] J. Millen and F. Muller. Cryptographic protocol generation from CAPSL. Technical Report SRI-CSL-01-07, SRI International, 2001.

[30] S. Mödersheim. *Models and Methods for the Automated Analysis of Security Protocols*. PhD Thesis, ETH Zurich, 2007. ETH Dissertation No. 17013.

[31] 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.

[32] D. v. Oheimb and J. Cuellar. Designing and verifying core protocols for location privacy. In *Information Security*, LNCS 4176. Springer-Verlag, 2006.

[33] C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner. Cryptographically Sound Theorem Proving. In *Proceedings of CSFW 19*, pages 153–166. IEEE Computer Society Press, 2006.

Appendix

Theorem 4. *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 $S_1 \sim S_2$.*

Theorem 4: We prove the statement by structural induction over transitions. The initial states of the two models are identical modulo $\sim$, since they contain no channel facts and the initial knowledge of the CCM material does not matter. Consider two states S_1 and S_2 of corresponding ICM and CCM specifications, respectively, with $S_1 \sim S_2$. Consider an ICM rule r that is applicable to S_1, producing S_1'. We show that from S_2 a state S_2' is reachable in the CCM with $S_1' \sim S_2'$.

The first case is that r is the rule of an honest agent. By the induction hypothesis, the corresponding CCM rule must be applicable to S_2' with respect to the corresponding facts/intruder knowledge and under the same matching substitution. Thus, the successor states S_1' and S_2' are in the same relation.

The second case is that r is a normal (not channel related) rule of the intruder. Also in this case the rule can similarly be applied in S_2 so that we have again equivalent successor states $S_1 \sim S_2$.

In the remaining cases r is one of the channel intruder rules, and we will use the assumption that the intruder knows all agent names, public-keys/pseudonyms, and tags without further mention:

Let $r = \mathsf{iknows}(B).\mathsf{iknows}(M).\mathsf{dishonest}(A) \Rightarrow \mathsf{athCh}_{A,B}(M)$. Since $S_1 \sim S_2$ and $\mathsf{dishonest}(A)$, the intruder knows M also in S_2. The encoding of the channel fact is either $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ if A is an agent name, or $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(A)}$ if A is a pseudonym. In either case, the intruder can generate this term, since A is dishonest and the intruder thus knows $\mathsf{inv}(\mathsf{ak}(A))$ or $\mathsf{inv}(A)$, respectively. Thus, an S_2' with $S_2' \sim S_1'$ is reachable.

Let $r = \mathsf{athCh}_{A,B}(M) \Rightarrow \mathsf{iknows}(M)$. Since $S_1 \sim S_2$, S_2 contains the encoding of the ideal channel fact, which is either $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ if A is an agent name or $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(A)}$ if A is a pseudonym. In either case, the intruder can obtain M by decryption (as he knows the agent name or the pseudonym B already by assumption), therefore an S_2' with $S_2' \sim S_1'$ is reachable.

Let $r = \mathsf{iknows}(B).\mathsf{iknows}(M) \Rightarrow \mathsf{cnfCh}_B(M)$. Since $S_1 \sim S_2$, the intruder knows M also in S_2. Therefore, he can generate the encoding of the ideal channel fact, which is either $\{\mathsf{ctag}, M\}_{\mathsf{ck}(B)}$ if B is an agent name or $\{\mathsf{ctag}, M\}_B$ if B is pseudonym, so an S_2' with $S_2' \sim S_1'$ is reachable.

Let $r = \mathsf{cnfCh}_B(M).\mathsf{dishonest}(B) \Rightarrow \mathsf{iknows}(M)$. Since $S_1 \sim S_2$, S_2 also contains the encoding of the ideal channel fact which is either $\{\mathsf{ctag}, M\}_{\mathsf{ck}(B)}$ if B is an agent name, or $\{\mathsf{ctag}, M\}_B$ if B is a pseudonym. Since B is dishonest, the intruder knows $\mathsf{inv}(\mathsf{ck}(B))$ or $\mathsf{inv}(B)$, respectively, and can thus obtain M. Therefore, an S_2' with $S_2' \sim S_1'$ is reachable.

Let $r = \mathsf{iknows}(B).\mathsf{iknows}(M).\mathsf{dishonest}(A) \Rightarrow \mathsf{secCh}_{A,B}(M)$. Since $S_1 \sim S_2$, the intruder knows M also in S_2. The encoding of the channel fact is one of the following: $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$ if A and B are agent names, $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_{\mathsf{ck}(B)}$ if A is a pseudonym and B is an agent name, $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_B$ if A is an agent name and B is a pseudonym, or $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_B$ if A and B are pseudonyms. Since A is dishonest, the intruder knows $\mathsf{inv}(\mathsf{ak}(A))$ or $\mathsf{inv}(A)$, respectively, and can thus generate the corresponding encoding of the ideal channel fact. Therefore, an S_2' with $S_2' \sim S_1'$ is reachable.

Let $r = \mathsf{secCh}_{A,B}(M).\mathsf{dishonest}(B) \Rightarrow \mathsf{iknows}(M)$. Since $S_1 \sim S_2$, S_2 contains the encoding of the ideal channel fact, which is one of the following: $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$ if A and B are agent names, $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_{\mathsf{ck}(B)}$ if A is a pseudonym and B is an agent name, $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_B$ if A is an agent name and B is a pseudonym, or $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_B$ if A and B are pseudonyms. Since B is dishonest, the intruder knows $\mathsf{inv}(\mathsf{ck}(B))$ or $\mathsf{inv}(B)$, respectively, and can thus decrypt the message to obtain M (recall that he knows all public keys and can thus decrypt the signature of A, too). Therefore, an S_2' with $S_2' \sim S_1'$ is reachable. □

Theorem 5. *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 $S_1 \sim S_2$.*

Theorem 5: As explained in § 4.5, we assume a well-typed attack in the CCM, and show that it can be simulated in the ICM. Note that, in the CCM, the intruder can arbitrarily compose messages using CCM material. Most of these “abuses” are already prevented by the assumption that the attack is typed and the use of tags in the encoding of channels. For instance, the intruder cannot use an encryption that results from a channel-encoding elsewhere in the protocol because of the tags. We first show that we can replace all abuses of CCM material with other keys and tags that the intruder knows and that he has also in the ICM. The second part of the proof is then to find a simulation for the remaining attack where CCM material is only used in a proper way for encoding channels.

Removing abuses. The rules of honest agents do not use CCM material in any other position than for the channels itself. Due to full decryption and typing, for a transition of an honest agent, the match between rule and current state must thus have the property that each key or tag matches directly a rule variable of the appropriate type. It is thus possible to uniformly replace the CCM material with different values throughout the attack in those places they are not used for the crypto-encoding of channels or as pseudonyms.

So for all messages that the intruder constructs using CCM material, let the intruder additionally construct analogous messages with other fresh keys and tags that he can generate, say a fresh pair $(ckt, \mathsf{inv}(ckt))$ for each $\mathsf{ck}(t)$ etc. Every agent will accept the changed incoming messages as well, replacing a variable for a public key K with ckt rather than $\mathsf{ck}(t)$ throughout its execution. Observe that the trace may contain some transitions of the intruder or honest agents that are unnecessary, i.e. the attack still works when removing them, e.g. the intruder composes a term that he never uses again and that is not relevant for a secrecy goal. It is without loss of generality to assume the attack does not contain any such unnecessary steps. We make one exception, however, namely we allow the decryption of encrypted messages and the projection of pair-components even if they are not used any further.

Observe also that honest agents will never give out private channel keys (due to the definition of the CCM), i.e. the intruder never finds out $\mathsf{inv}(\mathsf{ak}(A))$ or $\mathsf{inv}(\mathsf{ck}(A))$ for any honest agent A or $\mathsf{inv}(A)$ for any honest pseudonym A.

Translating the attack. So we have now an attack where all messages that honest agents deal with have CCM material only in those places that encode the channels in the CCM or for pseudonyms. Together with the assumption that there are no steps that are irrelevant to the attack, the intruder does not compose any messages with CCM material unless they are indeed sent to honest agents.

We now show that we can translate this attack transition by transition. Consider a state S_1 reached by a prefix of the attack in the CCM and a state S_2 reachable in the ICM with $S_1 \sim S_2$, and let S_1' be the state that is reached in the CCM by the next transition. We show how to reach a state S_2' in the ICM with $S_1' \sim S_2'$. We distinguish the following 12 transitions:

(i) The intruder composes a message where the result has the form $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ where A is an agent name. This requires that the intruder knows $\mathsf{inv}(\mathsf{ak}(A))$ and M in S_1. This can only work for $\mathsf{dishonest}(A)$, since, as noted before, the intruder cannot know $\mathsf{inv}(\mathsf{ak}(A))$ otherwise. Thus, by the rule (3), the intruder can generate $\mathsf{athCh}_{A,B}(M)$ in S_2. Hence, we reach an S_2' with $S_2' \sim S_1'$.

(ii) The intruder composes a message where the result has the form $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(A)}$ where A is a pseudonym. This requires that the intruder knows $\mathsf{inv}(A)$ and M in S_1, which in turn requires $\mathsf{dishonest}(A)$, because honest private keys are never leaked by the construction. Thus, by the rule (3), the intruder can generate $\mathsf{athCh}_{A,B}(M)$ in S_2. Hence, we reach an S_2' with $S_2' \sim S_1'$.

(iii) The intruder composes a message where the result has the form $\{\mathsf{ctag}, M\}_{\mathsf{ck}(B)}$ for an agent name B or $\{\mathsf{ctag}, M\}_B$ for a pseudonym B. To that end, he must know M also in S_2. We can thus apply rule (5) to obtain $\mathsf{cnfCh}_B(M)$ and thus reach an S_2' with $S_2' \sim S_1'$.

(iv) The intruder composes a message where the result has the form $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$ for agent names A and B. To that end, he must know $\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ first. We can exclude the case that this message was generated by an honest agent: if A is honest, then the message was sent by A as $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$ and the intruder must have analyzed the outermost encryption with $\mathsf{ck}(B)$ himself, i.e. B is dishonest, as an honest agent never sends out $\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$. But that in turn means that the intruder re-encrypts a message that he already has in encrypted form and that can only be received by himself anyway. So A is dishonest. Also, M is in S_2, and thus we obtain the fact $\mathsf{secCh}_{A,B}(M)$ by rule (7), and hence we reach an S_2' with $S_2' \sim S_1'$.

(v) We proceed similarly for the cases of a secure channel message where one or both of the two ends are pseudonymous, i.e. for generating $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_{\mathsf{ck}(B)}$ where A is a pseudonym and B is an agent name, $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_B$ where A is an agent name and B is a pseudonym, and $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_B$ where both A and B are pseudonyms, we can derive again that A is dishonest and thus the intruder can generate $\mathsf{secCh}_{A,B}(M)$ by rule (7).

(vi) The intruder composes any message that is not covered by the previous cases. If it does not contain any CCM material, then this can similarly be done in the ICM. Otherwise, the resulting message is irrelevant for $\sim$ anyway and we can proceed with $S_2' = S_2$.

(vii) The intruder analyzes a message of the form $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}$ where A is an agent name or $\{\mathsf{atag}, B, M\}_{\mathsf{inv}(A)}$ where A is a pseudonym, and by analyzing the resulting pair obtains M. By the induction hypothesis, S_2 contains $\mathsf{athCh}_{A,B}(M)$ and by the rule (4) and he can obtain M. Hence we reach an S_2' with $S_2' \sim S_1'$.

(viii) The intruder analyzes a message of the form $\{\mathsf{ctag}, M\}_{\mathsf{ck}(B)}$ where B is an agent name or $\{\mathsf{ctag}, M\}_B$ where B is a pseudonym, obtaining M by further analysis of the resulting pair. He can only do this analysis if B is dishonest, i.e. he has $\mathsf{inv}(\mathsf{ck}(B))$ or $\mathsf{inv}(B)$, respectively. By the induction hypothesis, S_2 contains $\mathsf{cnfCh}_B(M)$, thus by rule (6), the intruder can obtain M. Hence we reach an S_2' with $S_2' \sim S_1'$.

(ix) The intruder analyzes a message of the form $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_{\mathsf{ck}(B)}$ or $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_{\mathsf{ck}(B)}$ where A is an agent name or pseudonym, respectively, and B is an agent name. This is only possible if he knows $\mathsf{inv}(\mathsf{ck}(B))$, thus B is dishonest, and he can proceed to obtain M. By the induction hypothesis, $\mathsf{secCh}_{A,B}(M)$ is contained in S_2, and thus by (8), the intruder can obtain M here as well. Hence we reach an S_2' with $S_2' \sim S_1'$.

(x) We proceed similarly for the two remaining cases, i.e. for analyzing $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(\mathsf{ak}(A))}\}_B$ where A is an agent name and B is a pseudonym, and $\{\{\mathsf{stag}, B, M\}_{\mathsf{inv}(A)}\}_B$ where both A and B are pseudonyms. Again, B must be dishonest, and thus the intruder can obtain M. By the induction hypothesis, $\mathsf{secCh}_{A,B}(M)$ is contained in S_2, and thus by (8), the intruder can obtain M here as well. Hence we reach an S_2' with $S_2' \sim S_1'$.

(xi) The intruder analyzes any message that is not covered by the previous cases. If the analyzed message contains CCM material, then either it is the superfluous analysis of a message that the intruder composed himself or a subsequent one of the previous cases, i.e. already covered. Otherwise,

the decomposition works identically in the ICM.

(xii) A transition of an honest agent. Here the incoming message is either in the intruder knowledge in both S_1 and S_2, if it is on an insecure channel, or it is a secure channel, then the respective channel fact is contained in S_2 by $S_1 \sim S_2$. Similarly, the outgoing message is either in both models directly in the intruder knowledge (for an insecure channel) or the respective ideal channel fact is in S_2.

Thus we have transformed the CCM attack trace to one of the ICM. □

Theorem 6. *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 $P_2[P_1]$.*

Theorem 6: We prove this for each kind of channel separately, but the main idea is the same: we assume an attack trace against $Par(P_2[P_1])$ and show that there is also an attack against the vertical or the horizontal composition of P_1 and P_2. Due to compositionality, there is thus an attack against one of the protocols P_1, P_2, or P_1^* individually, contradicting the assumptions.

Authentic Channels. Consider an attack against $Par(P_2[P_1])$. We consider transitions related to the channel realized by P_1, i.e. transitions of P_2 where an agent sends or receives on the channel. If there are no such steps at all, then the attack works on P_2 already, since no protocol execution in the attack even got to the part that is implemented by P_1.

For any transition that corresponds to sending on the channel realized by P_1 by an honest agent or pseudonym a, we have a corresponding witness event $\mathsf{witness}(a, b, P_1, m)$ of P_1 for the concrete payload message m sent to b. Similarly, whenever an honest agent or pseudonym b receives from the channel, we have a request event $\mathsf{request}(a, b, P_1, m)$ for every b who thinks to have received m from a (independent of whether a is honest or not).

We first consider what happens if there is a request event for an honest receiver without a corresponding witness event and show that in this case either P_1^* or P_2 breaks.

Consider the case that in the attack trace an event $\mathsf{request}(a, b, P_1, m)$ occurs where b is honest (since only honest receivers generate this event) and either (i) a is honest and there is no corresponding $\mathsf{witness}(a, b, P_1, m)$, or (ii) a is dishonest and the intruder does not know m. Thus, the authentic transmission of m on the channel is violated. If there are several such violations in the attack trace, let us consider the first one, i.e. a $\mathsf{request}(\cdot)$ event that breaks and all other before are fine.

Up to this point, for every $\mathsf{witness}(a, b, P_1, m)$ on the attack trace, we also add the event $\mathsf{athCh}_{a,b}(m)$, which would have been generated if we had considered the "ideal" P_2 instead of the more concrete $P_2[P_1]$. For every $\mathsf{request}(a, b, P_1, m)$ with a dishonest a where $\mathsf{iknows}(m)$ holds (at that point), we similarly add $\mathsf{athCh}_{a,b}(m)$. This can be constructed by the intruder since he knows m and b using rule (3). Thus the receipt of messages in the ideal P_2 is possible for every $\mathsf{request}(a, b, P_1, m)$ of the concrete $P_2[P_1]$ but the last one. Thus, inductively all steps of $P_2[P_1]$ in the trace also work for P_2 using $\mathsf{athCh}_{a,b}(m)$ up to the attack state. So we have an attack against the horizontal composition of P_1^*, and P_2, which was to show.

This concludes the case that the attack trace has requests without corresponding witness events or $\mathsf{iknows}(m)$ problems. Now let us consider that every request has a corresponding witness or both the sender is dishonest and $\mathsf{iknows}(m)$. Then by the same construction as above, namely adding $\mathsf{athCh}_{a,b}(m)$ in the respective places, we can conclude that the attack trace would work on the abstract P_2 in parallel with P_1, i.e. is an attack against the horizontal composition of P_1^* and P_2.

Confidential Channels. This case is proven similarly. When we have no whisper and hear events of P_1 at all, then we simply have not used the channel and thus there is an attack against P_2 already.

Otherwise, for every $\mathsf{whisper}(b, P_1, m)$ event (which must have been generated by an honest sender), we add also $\mathsf{cnfCh}_b(m)$.

For every $\mathsf{hear}(b, P_1, m)$ (where b must be honest) if $\mathsf{whisper}(b, P_1, m)$ did not occur before, m was supposedly generated by a dishonest sender. Thus, if indeed $\mathsf{iknows}(m)$ holds at that point, we add $\mathsf{cnfCh}_b(m)$ (which is possible due to (5)). Otherwise, if $\mathsf{iknows}(m)$ does not hold there, we already have an attack against P_1. We can thus ensure that all channel-related actions work on the ideal P_2 as before, so an ideal channel receive using $\mathsf{cnfCh}_b(m)$ is always possible. So up to the attack, the trace similarly works for the horizontal composition of P_1^* and P_2.

Secure Channels. Here we have again a similar construction. When an honest agent sends a message on the secure channel realized by P_1, we have the events $\mathsf{witness}(a, b, P_1, m)$ and $\mathsf{whisper}(b, P_1, m)$; we add for this pair of events the event $\mathsf{secCh}_{a,b}(m)$. Moreover, when b receives m from a dishonest sender, i.e. when we have $\mathsf{request}(a, b, P_1, m)$ and $\mathsf{hear}(b, P_1, m)$ for a dishonest a, then either $\mathsf{iknows}(m)$ at that point and we can add the event $\mathsf{secCh}_{a,b}(m)$, or we have an attack already due to the request requirements. So, as long as every $\mathsf{request}(a, b, P_1, m)$ is either preceded by a $\mathsf{witness}(a, b, P_1, m)$ or a is dishonest and $\mathsf{iknows}(m)$ (otherwise we have an attack), then the receiving works in the ideal P_2, as $\mathsf{secCh}_{a,b}(m)$ can be added to the trace—so we have reached an attack trace for the horizontal composition of P_1^* and P_2. □