Secure Refinements of Communication Channels

Vincent Cheval[1], Véronique Cortier[2], and Eric le Morvan[2]

1 **School of Computing, University of Kent, UK & LORIA, INRIA, France**
2 **LORIA, CNRS, France**

Abstract

It is a common practice to design a protocol (say Q) assuming some secure channels. Then the secure channels are implemented using any standard protocol, e.g. TLS. In this paper, we study when such a practice is indeed secure.

We provide a characterization of both confidential and authenticated channels. As an application, we study several protocols of the literature including TLS and BAC protocols. Thanks to our result, we can consider a larger number of sessions when analyzing complex protocols resulting from explicit implementation of the secure channels of some more abstract protocol Q.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Protocol, Composition, Formal methods, Channels, Implementation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.575

1 Introduction

When designing a protocol, it is common to assume a secure, confidential, or authentic channel. Authentic channels may be read but not written in. Symmetrically, confidential channels may be written in but not read. Secure channels are both authentic and confidential. For example, payment protocols like 3D-secure are supposed to be run over a secure channel such as TLS. Similarly, many services such as public key registration assume an authenticated channel. How to implement these secure channels is left unspecified and, intuitively, the security of a payment protocol should not depend on the particular choice of implementation of its secure channels. A typical example of a generic realization of a secure channel is TLS. For authentication, one usually relies on a password-based authentication or on previously established keys (used e.g. for signature or MACs). Is it safe to use these protocols in any context? What is a secure or authenticated channel? This paper aims at characterizing channels that have security properties. For example, assume Q is a secure protocol (e.g. a payment protocol) that requires a secure channel. Which properties should a protocol P achieve in order to securely realize the secure channels of Q? These properties should of course be independent of Q since P and Q are typically designed in totally independent contexts. In the remaining of this introduction, Q will refer to the "main" protocol while P will refer to a protocol realizing secure channels (for several notions of security).

Our contributions. Our first contribution is a characterization of both secure, confidential, and authenticated channels. We actually characterize what it means for a channel to be readable or not, and writable or not. Then the realization of a secure channel typically proceeds in two phases. First, some values are established by the protocol P, for example short-term symmetric keys or MAC keys. Quite unsurprisingly, we show that these values need to be secret and appropriately shared. Then the messages of Q are transported or *encapsulated* using the values established by P. For example, the messages of Q may be

encrypted with a key established by P. We provide a characterization of secure encapsulations both for secure, confidential, and authentic channels. A key feature of our characterization is that it is independent of P and Q, which allows for a modular analysis. We show that standard encapsulations (e.g. typical use of encryption, signatures, or MACs) enjoy the requested properties.

Our second and main contribution is to show how to securely compose protocols. Intuitively, our main result guarantees that whenever P is a secure key exchange protocol and $\mathcal{E}$ is a secure encapsulation then $P \cdot^{\mathcal{E}} Q$ is as secure as Q where $P \cdot^{\mathcal{E}} Q$ denotes the protocol obtained from Q by implementing its secure channels using P and $\mathcal{E}$. The interest of our result is twofolds. First, it provides foundational grounds to a common practice where protocols are typically designed and studied independently and then combined. We show that such a practice is actually secure under reasonably light assumptions: primitives shared between P, $\mathcal{E}$, and Q should be tagged as proposed in [4]. Tagging is a standard practice that avoids message confusion. Second, our result provides a technique for analyzing a complex protocol: it is sufficient to analyse its components to deduce security of the whole protocol. To express and prove our result, we have developed a framework, an extension of the applied-pi calculus [2], that allows to easily talk about protocols roles and sessions, a missing aspect in the applied-pi calculus. To illustrate our approach, we show that TLS is a secure implementation of secure channels. Similarly we show that the BAC protocol [1] is also a secure implementation of a secure channel and may be safely used with the Passive Authentication (PA) protocol as prescribed for the biometric passport [1]. Using the CL-Atse tool [18], we analyse several combined protocols. Thanks to our combination result, it is possible to analyse protocols in isolation which allows to consider a larger number of sessions.

Related work. One seminal work on composition is the one of Guttman and Thayer [13]. They show that two protocols can be composed without one damaging the security of the other as soon as they are "independent". However, this independence notion needs to be checked for any protocol execution and cannot be statically checked at the protocol specification level. Later, Guttman [11] provides a criterion on the specification of P and Q such that P can be safely composed with Q. Intuitively, Q should not break some invariant satisfied by P and conversely. While the work of [11] focuses on authentication and secrecy properties, [12] more generally devises a framework for defining protocol goals and designing, step by step, protocols that fulfill them. In [10], the strand space model is used in a modular way, to analyse protocols components by components. The disjunction criteria cannot be checked statically. All these approaches provide a framework that allows to reason modularly when analysing the combination of two protocols P and Q, typically expressing invariants satisfied by P that are shown sufficient to prove security of Q. This simplifies the proof of P combined with Q but requires the knowledge of both protocols. Compared to our work, we propose a criteria for a protocol P to securely implement a secure channel, independently of the protocol Q that will use it (provided primitives are tagged).

Under tagging assumptions similar to ours, it was already shown that P and Q can be safely run in parallel even if they share long-term keys [7]. In passing, we generalize this result to the case where long-term keys may be used as payload. [6] explains when two protocols may be used sequentially, with Q using data established by P. The main difference with our work is that messages may not be transformed when composing protocols. Therefore, [7, 6] cannot be used to (securely) implement abstract channels. Note also that [6] may not consider compromised sessions, that is sessions between honest and dishonest agents. The problem we address here is referred to as *sequential composition* in [16], where the messages of Q are used

as payloads in the composed protocol $P \cdot^{\mathcal{E}} Q$. [16] provides a nice exposition of the generic problem of a protocol Q using a protocol P as subprotocol and lists sufficient (semantical) conditions for combining two protocols. These conditions require again the knowledge of both P and Q. Datta et al. (e.g. [8]) have also studied secure protocol composition in a broader sense: protocols can be composed in parallel, sequentially or protocols may use other protocols as components. However, they do not provide any syntactic conditions for a protocol P to be safely executed in parallel with other protocols. For any protocol P' that might be executed in parallel, they have to prove that the two protocols P and P' satisfy each other invariants. Their approach is thus rather designed for component based design of protocols.

2 Model

Our model is inspired from the applied-pi calculus [2], extended to an explicit notion of roles and agents.

2.1 Messages

Messages are modeled using a typed term algebra. We assume an infinite set of names $\mathcal{N} = \mathcal{N}_D \uplus \mathcal{N}_H$ of *base type* and a set $\mathcal{C}h$ of names of *channel type*. The set $\mathcal{N}_H$ (resp. $\mathcal{N}_D$) represents the names accessible by honest (resp. dishonest) agents. We also a consider an infinite set of variables $\mathcal{X}$ and a finite signature $\mathcal{F}$ of function symbols operating and returning terms of *base type*. More precisely, we consider $\mathcal{F} = \mathcal{F}_c \uplus \mathcal{F}_{cst} \uplus \mathcal{F}_{key}$ where $\mathcal{F}_{cst}$ contains only constants, all functions in $\mathcal{F}_{key}$ are unary, and $\mathcal{F}_c = \{\langle\,\rangle/2, \mathsf{f}_1/n_1, \ldots, \mathsf{f}_k/n_k\}$ contains the binary function symbol $\langle\,\rangle$ used to denote concatenation and other function symbols f_i of arity n_i. Terms are defined as names, variables and function symbols applied to other terms. The set of terms built from $N \subseteq \mathcal{N} \cup \mathcal{C}h$, $X \subseteq \mathcal{X}$ and by applying the function symbols in $F \subseteq \mathcal{F}$ is denoted by $\mathcal{T}(F, N \cup X)$. We denote by $st(t)$ the set of subterms of t. We denote by $vars(t)$ (resp. $names(t)$) the set of variables (resp. names) in t. When $vars(t) = \emptyset$, we say that t is *ground*. To represent events that may occur during a protocol execution, we assume an infinite signature $\mathcal{E}v$ distinct from $\mathcal{F}$. We say that a term $e(t_1, \ldots, t_n)$ with $e \in \mathcal{E}v$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X})$ is an event.

▶ **Example 1.** A standard signature to represent encryption and signature is $\mathcal{F}_{std}$, the signature built from a finite set of constants, functions $\mathcal{F}_{cstd} = \{\mathsf{senc}/2, \mathsf{aenc}/2, \mathsf{sign}/2, \mathsf{h}/1, \langle\rangle/2\}$ and $\mathcal{F}_{kstd} = \{\mathsf{pk}/1, \mathsf{vk}/1\}$. The function symbol senc (resp. aenc) represents the symmetric (resp. asymmetric) encryption. We denote by $\mathsf{pk}(s)$ the public key associated s. The function symbol sign represents the digital signature where $\mathsf{vk}(s)$ is the verification associated to s. We write $\langle u, v\rangle$ as syntactic sugar for $\langle\rangle(u, v)$.

We model the algebraic properties of the cryptographic primitives by a set of inference rules $\mathcal{I}$ composed of composition and decomposition rule described as follows:

$$\frac{x_1 \quad \ldots \quad x_k}{\mathsf{f}(x_1, \ldots, x_k)}\ \mathsf{f}\text{-comp} \qquad \frac{\langle x_1, x_2\rangle}{x_1} \qquad \frac{\langle x_1, x_2\rangle}{x_2}$$

$$\frac{\mathsf{f}(x, u_1, \ldots, u_n) \quad v_1 \quad \ldots \quad v_m}{x}\ \mathsf{f}\text{-decomp}$$

where for all $j \in \{1, \ldots, n\}$, for all $k \in \{1, \ldots, m\}$, $u_j, v_k \in \mathcal{T}(\mathcal{F}_{key}, \mathcal{X})$ and $vars(v_1, \ldots, v_k) \subseteq \{u_1, \ldots, u_n, x\}$. For each $\mathsf{f} \in \mathcal{F}$, the set $\mathcal{I}$ contains a unique f-comp rule and there is no f-decomp rule when $\mathsf{f} \in \mathcal{F}_{key}$. Given a set or sequence of terms S and a term t, the deducibility relation is inductively defined as follows. The term t is deducible from S, denoted

$S \vdash t$, when $t \in S \cup \mathcal{F}_{cst} \cup \mathcal{N}_D$ or there exists a substitution σ and an inference rule in $\mathcal{I}$ with premisses $u_1, \ldots, u_n$ and conclusion u such that $t = u\sigma$ and for all $i \in \{1, \ldots, n\}$, $S \vdash u_i\sigma$.

▶ **Example 2.** Continuing Example 1, we define the set $\mathcal{I}_{std}$ of decomposition rules as follows.

$$\frac{\mathsf{senc}(x,y) \quad y}{x} \qquad \frac{\mathsf{aenc}(x,\mathsf{pk}(y)) \quad y}{x} \qquad \frac{\mathsf{sign}(x,y) \quad \mathsf{vk}(y)}{x} \qquad \frac{\langle x,y\rangle}{x} \qquad \frac{\langle x,y\rangle}{y}$$

We have that $\mathsf{senc}(\langle a,c\rangle,k), k \vdash a$ but $\mathsf{aenc}(\langle a,c\rangle,\mathsf{pk}(k)),\mathsf{pk}(k) \nvdash a$.

2.2 Agents

In standard process algebra (*e.g.* [2]), the notion of agents is usually implicit. Typically, a process that models the behavior of the different honest agents is a single process where all agents are implicitly represented. However, to model protocol composition, we need to explain how to compose each role and thus we need to talk about each agent separately. Therefore, we explicit the presence of agents in our model. Interestingly, our model may also be used to specify semi-honest agents which may directly communicate with the attacker during the protocol execution, still hiding some secrets from him. We consider an infinite set of agents $\mathcal{A}gt = \{A, B, \ldots\} = \mathcal{A}gt_H \uplus \mathcal{A}gt_D$ where $\mathcal{A}gt_H$ and $\mathcal{A}gt_D$ represent respectively honest and dishonest agents. Each agent possesses private data such as keys. Thus, we consider $\mathcal{N}_{\mathcal{A}gt}$ a subset of $\mathcal{N}$ as an infinite partition $\mathcal{N}_{\mathcal{A}gt} = \biguplus_{A \in \mathcal{A}gt} \mathcal{N}_A$ where $\mathcal{N}_A$ intuitively are the names accessible by the agent A. By convention, $k[A]$ denotes a name in $\mathcal{N}_A$.

2.3 Protocols

In the spirit of [2], we model protocols through a process algebra. We represent explicitly confidential, secure, and authenticated channels. Formally, we partition the set of channels into three infinite sets $\mathcal{C}h = \mathcal{C}h_a \uplus \mathcal{C}h_c \uplus \mathcal{C}h_s \uplus \mathcal{C}h_p$ where $\mathcal{C}h_a, \mathcal{C}h_c, \mathcal{C}h_s, \mathcal{C}h_p$ respectively represent the sets of authenticated, confidential, secure and public channels. The syntax of our calculus is as follows:

Roles of agent A

$$R_A, R'_A := \quad 0 \mid \mathsf{out}_A(c,u).R_A \mid \mathsf{in}_A(c,v).R_A \mid \mathsf{new}\ k.R_A \mid \mathsf{event}_A(ev).R_A$$

Channel and agent declarations

$$C, C' := \quad R_A \mid \mathsf{new}_{ta}\ c.C \mid C \mid C'$$

Processes

$$P, Q := \quad C \mid P \mid Q \mid !P \mid \mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub}, \mathcal{K}_{prv}).P$$

where $c \in \mathcal{C}h$, $A \in \mathcal{A}gt$, ta is the tuple of agents in C such that c occurs in their role, k is name, u and v are terms, ev is an event, $\mathcal{K}_{pub}$ and $\mathcal{K}_{prv}$ are sets of ground terms with $names(\mathcal{K}_{pub}) \subseteq \mathcal{N}_A$, $names(\mathcal{K}_{prv}) \subseteq \mathcal{N}_{\mathcal{A}gt}$ and $\mathcal{A} \subseteq \mathcal{A}gt$.

The behavior of an agent A is described in a *role* R_A that consists of a sequence of inputs, outputs, creations of names and emissions of events. The role $\mathsf{out}_A(c,u).R_A$ outputs the term u on the channel c and then behaves like R_A. The role $\mathsf{in}_A(c,v).R_A$ inputs a message from channel c and expects it to be an instance of v. The role $\mathsf{new}\ k.R_A$ generates a fresh name k. Processes express how the roles of different agents are combined. The process $\mathsf{new}_{ta}\ c$ allocates an abstract channel to the agents in ta. The process $P \mid Q$ expresses the parallel execution of P and Q. The process $!P$ represents the replication of P. The process $\mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub}, \mathcal{K}_{prv}).P$ selects a new agent A amongst $\mathcal{A}$. The set $\mathcal{K}_{pub}$ typically indicates the public keys of A while $\mathcal{K}_{prv}$ contains the (secret) long term keys known by A. The variables in a role are uniquely bound by the first input in which they appear. The

$(P \mid \mathsf{out}_A(c,u).R_A, \Phi, \mu, \theta) \to (P \mid R_A, \Phi', \mu', \theta)$ OUT
with $\Phi' = \Phi$ if $c \in \mathcal{C}h_c \cup \mathcal{C}h_s$ else $\Phi' = \Phi \cdot [u]$ and $\mu' = \mathsf{rect}(c,u,\mu)$ if $c \notin \mathcal{C}h_p$ else $\mu' = \mu$

$(P \mid \mathsf{in}_A(c,v).R_A, \Phi, \mu, \theta) \to (P \mid R_A\sigma, \Phi, \mu, \theta)$ IN
if $\exists\sigma$ s.t. $dom(\sigma) = vars(v)$ and either $v\sigma \in c\mu$ or else $c \in \mathcal{C}h_p \cup \mathcal{C}h_c$ and $\Phi \vdash v\sigma$

$(P \mid \mathsf{new}\ k.R_A, \Phi, \mu, \theta) \to (P \mid R_A\{^{k'}/_k\}, \Phi, \mu, \theta)$ NEW-K
with k' fresh in $\mathcal{N}_H$ if $A \in \mathcal{A}gt_H$ else $k' \in \mathcal{N}_D$

$(P \mid \mathsf{new}_{ta}\ c.C[R_{A_1}, \ldots, R_{A_n}], \Phi, \mu, \theta) \to (P \mid [R'_{A_1}, \ldots, R'_{A_n}], \Phi, \mu, \theta')$ NEW-C
$\forall i, R'_{A_i} = R_{A_i}$ if $c \notin ch(R_{A_i})$ else $R'_{A_i} = R_{A_i}\{^{c_{A_i}}/_c\}$ with $c_{A_i} \in \mathcal{C}h_p$ if $ta \cap \mathcal{A}gt_D \neq \emptyset$ else $c_{A_i} \in S \cup \bigcup_{B \in ta} \theta(c,B,ta) \setminus \theta(c,A_i,ta)$ and $S \subseteq \mathcal{C}h_a$ fresh (resp. $\mathcal{C}h_c$, $\mathcal{C}h_s$) if $c \in \mathcal{C}h_a$ (resp. $\mathcal{C}h_c$, $\mathcal{C}h_s$). Moreover, $\theta = \theta'$ if $ta \cap \mathcal{A}gt_D \neq \emptyset$ else $\theta' = \mathsf{recc}(\{(c_A, A)\}_{A \in ta}, ta, c, \theta)$.

$(P \mid !Q, \Phi, \mu, \theta) \to (P \mid !Q \mid Q\rho, \Phi, \mu, \theta)$ REPL
with ρ a fresh renaming of $vars(Q)$

$(P \mid \mathsf{event}_A(ev).R, \Phi, \mu, \theta) \xrightarrow{ev} (P \mid R, \Phi, \mu, \theta)$ EVENT

$(P \mid \mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub}, \mathcal{K}_{prv}).Q, \Phi, \mu, \theta) \to (P \mid Q\sigma, \Phi \cdot S, \mu, \theta)$ AGENT
with $\sigma = \{^{A'}/_A\}$, $A' \notin fa(Q)$, $S = \mathcal{K}_{pub}\sigma$ if $A' \in \mathcal{A} \cap \mathcal{A}gt_H$ else $S = \mathcal{K}_{pub}\sigma \cdot \mathcal{K}_{prv}\sigma$

Figure 1 Semantics of configuration.

channels are bound by the operators new . The agents in a process are also bound by agent creation. In a protocol, we assume that a name or variable is syntactically bound only once. A variable (resp. agent, channel) that is not bound in P is free. We denote by $fa(P)$, $ba(P)$, $fv(P)$, $bv(P)$, $fn(P)$ and $bn(P)$ the sets of free and bound agents, variables and names in P respectively. We say that P is closed when $fv(P) = \emptyset$. Given a process P and an agent A, we denote by $ch_A(P)$ the sets of channels that occur in the roles of A in P.

A role is executable if it only outputs terms that may be deduced from its inputs, the generated values (nonces and keys), and the long-term keys used in the role.

▶ **Definition 3.** Let $R_A = r_1.\ldots.r_n$ be a role of an agent A. We say that R_A is *executable* when for all $i \in \{1,\ldots,n\}$, if $r_i = \mathsf{out}_A(c,u)$ then $names(r_1,\ldots,r_i) \cup S \vdash u$ where $S = \{v \mid j < i \wedge (r_j = \mathsf{in}_A(d,v) \vee r_j = \mathsf{new}\ v)\}$. A process P is executable when all the roles in P are executable.

The state of a protocol during its execution is represented by a *configuration* (P, Φ, μ, θ) where P is a closed process, Φ is a sequence of ground terms representing the knowledge of the attacker, μ is a mapping from channels to sets of terms representing the messages sent over non-public channels and θ is a mapping from triplets of channel, agent, tuple of agents to sets of channels. The semantics is given in Figure 1. The rule OUT indicates that the attacker obtains messages on public or authenticated channels. In this rule, $\mathsf{rect}(c,t,\mu)$ is the mapping μ' where t was recorded as being sent over c. Formally, $\mu'(c') = \mu(c')$ for any $c' \neq c$ and $\mu'(c) = \mu(c) \cup \{t\}$. With rule IN the attacker can inject on c any message that he can deduce from his knowledge when c is a public or confidential channel. He can also relay any message that was previously sent on c. The rule NEW-K generates a fresh name of $\mathcal{N}_H$ or $\mathcal{N}_D$ depending on whether the agent A is honest or not. The rule NEW-C allocates to the role of an agent a channel possibly fresh or that has already been used by other roles in different sessions. In this rule, $\mathsf{recc}(S,ta,c,\theta)$ is the mapping θ in which we record the channels allocated to the agents. Formally, $\theta'(c',A',ta') = \theta(c',A',ta')$ for any $A' \notin ta'$ or

$(c', ta') \neq (c, ta)$, and $\theta'(c, A, ta) = \theta(c, A, ta) \cup \{d\}$ for any $(d, A) \in S$. The rule AGENT selects an agent from $\mathcal{A}$ and adds $\mathcal{K}_{pub}$ to the knowledge of the attacker. Additionally, if the agent is dishonest, the rules adds $\mathcal{K}_{prv}$. When $(P, \Phi, \mu, \theta) \xrightarrow{e_1} \ldots \xrightarrow{e_n} (P', \Phi', \mu', \theta')$, we write $(P, \Phi, \mu, \theta) \xrightarrow{e_1 \cdot \ldots \cdot e_n} (P', \Phi', \mu', \theta')$.

▶ **Example 4.** An electronic passport is a paper passport containing a RFID chip that stores the information printed on the passport. The protocols used to access these private data are specified in the International Civil Aviation Organization standard [1]. Before exchanging any private data, an electronic passport and a reader must establish session keys through a key-exchange protocol, called Basic Access Control (BAC), that prevents eavesdropping on further communication. The BAC protocol relies on two keys ke and km that are printed on the passport and thus can be obtained by the reader through optical scanning. We described below the BAC protocol, between a passport (P) and a reader (R). We assume encrypted messages to be tagged with a. The use of tagging will be explained later on.

$$\mathsf{R} \to \mathsf{P} : \mathsf{challenge}$$
$$\mathsf{P} \to \mathsf{R} : n_\mathsf{P}$$
$$\mathsf{R} \to \mathsf{P} : \langle \mathsf{senc}(\langle \mathsf{a}, n_\mathsf{R}, n_\mathsf{P}, k_\mathsf{R} \rangle, ke), \mathsf{mac}(\langle \mathsf{a}, \mathsf{senc}(\langle \mathsf{a}, n_\mathsf{R}, n_\mathsf{P}, k_\mathsf{R} \rangle, ke) \rangle, km) \rangle$$
$$\mathsf{P} \to \mathsf{R} : \langle \mathsf{senc}(\langle \mathsf{a}, n_\mathsf{P}, n_\mathsf{R}, k_\mathsf{P} \rangle, ke), \mathsf{mac}(\langle \mathsf{a}, \mathsf{senc}(\langle \mathsf{a}, n_\mathsf{P}, n_\mathsf{R}, k_\mathsf{P} \rangle, ke) \rangle, km) \rangle$$

After receiving a challenge command from the reader, the passport generates a fresh name n_P that will be used to verify the authenticity of the messages he will receive later on. Upon receiving n_P, the reader generates two nonces $n_\mathsf{R}, k_\mathsf{R}$ and sends back to the passport all three nonces encrypted with the key k_e and a mac with the key k_m. The nonce n_R has also an authenticity purpose whereas k_R will be the reader's contribution to the session keys. The passport then checks the mac using k_m and the cipher by decrypting it using k_e and verifying the presence of n_P in the plain text. If all verifications succeed, the passport generates a nonce k_P, the passport's contribution to the session keys, and sends it to the reader. At the end of the protocol, both reader and passport know k_R and k_P that they use to generate two session keys $\mathsf{f}_1(k_\mathsf{R}, k_\mathsf{P})$ and $\mathsf{f}_2(k_\mathsf{R}, k_\mathsf{P})$. In our syntax, the roles of the reader (R_R) and of the passport (R_P) can be expressed as follows.

$$\begin{aligned} R_\mathsf{P} = \; & \mathsf{in}_\mathsf{P}(c, \mathsf{challenge}).\mathsf{new}\ n_\mathsf{P}.\mathsf{out}_\mathsf{P}(c, n_\mathsf{P}).\mathsf{in}_\mathsf{P}(c, \langle M, \mathsf{mac}(\langle \mathsf{a}, M \rangle, km[\mathsf{P}]) \rangle). \\ & \mathsf{new}\ k_\mathsf{P}.\mathsf{out}_\mathsf{P}(c, \langle N, \mathsf{mac}(\langle \mathsf{a}, N \rangle, km[\mathsf{P}]) \rangle).0 \end{aligned}$$

$$\begin{aligned} R_\mathsf{R} = \; & \mathsf{out}_\mathsf{R}(c, \mathsf{challenge}).\mathsf{in}_\mathsf{R}(c, z).\mathsf{new}\ k_\mathsf{R}.\mathsf{new}\ n_\mathsf{R}.\mathsf{out}_\mathsf{R}(c, \langle U, \mathsf{mac}(\langle \mathsf{a}, U \rangle, km[\mathsf{P}]) \rangle). \\ & \mathsf{in}_\mathsf{R}(c, \langle V, \mathsf{mac}(\langle \mathsf{a}, V \rangle, km[\mathsf{P}]) \rangle).0 \end{aligned}$$

with $c \in \mathcal{C}h_p$, $M = \mathsf{senc}(\langle \mathsf{a}, x, n_\mathsf{P}, y \rangle, ke[\mathsf{P}])$, $N = \mathsf{senc}(\langle \mathsf{a}, n_\mathsf{P}, x, k_\mathsf{P} \rangle, ke[\mathsf{P}])$, $U = \mathsf{senc}(\langle \mathsf{a}, n_\mathsf{R}, z, k_\mathsf{R} \rangle, ke[\mathsf{P}])$ and $V = \mathsf{senc}(\langle \mathsf{a}, z, n_\mathsf{R}, w \rangle, ke[\mathsf{P}])$. An honest reader communicating with unbounded number of passports, possibly dishonest, can be modeled as the process:

$$BAC = \; \mathsf{ag}(\mathsf{R}, \{\mathsf{R}\}, \emptyset, \emptyset).!\mathsf{ag}(\mathsf{P}, \mathcal{P}, \emptyset, \{ke[\mathsf{P}], km[\mathsf{P}]\}).(R_\mathsf{P} \mid R_\mathsf{R})$$

where $\mathcal{P}$ is an infinite set of agents containing honest and dishonest agents and $\mathsf{R} \notin \mathcal{P}$. The following trace would correspond to the execution of a session with a dishonest passport I and a session of an honest one A both in $\mathcal{P}$.

$$\begin{aligned} (BAC, \emptyset, \emptyset, \emptyset) &\to^* (BAC \mid \mathsf{ag}(\mathsf{P}, \mathcal{P}, \emptyset, \{ke[\mathsf{P}], km[\mathsf{P}]\}).(R_\mathsf{P} \mid R_\mathsf{R}), \emptyset, \emptyset, \emptyset) \\ &\to (BAC \mid R_\mathsf{P}\sigma_A \mid R_\mathsf{R}\sigma_A, \emptyset, \emptyset, \emptyset) \\ &\to^* (BAC \mid R_\mathsf{P}\sigma_A \mid R_\mathsf{R}\sigma_A \mid R_\mathsf{P}\sigma_I \mid R_\mathsf{R}\sigma_I, [ke[\mathsf{I}], km[\mathsf{I}]], \emptyset, \emptyset) \\ &\to (BAC \mid R_\mathsf{P}\sigma_A \mid R_\mathsf{R}\sigma_A \mid R_\mathsf{P}\sigma_I \mid Q, [ke[\mathsf{I}], km[\mathsf{I}], \mathsf{challenge}], \emptyset, \emptyset) \\ &\to^* \ldots \end{aligned}$$

where $\mathsf{P}\sigma_A = \mathsf{A}$, $\mathsf{P}\sigma_I = \mathsf{I}$, $R_R\sigma_I = \mathsf{out}_\mathsf{I}(c, \mathsf{challenge}).Q$ and σ_A, σ_I are fresh renaming of bound variables. By convention, $\mu = \emptyset$ (resp. $\theta = \emptyset$) denotes the mapping that maps any argument to the emptyset: $\mu(c) = \emptyset$ (resp. $\theta(c, A, ta) = \emptyset$) for any c, A, ta.

3 Composition

In the previous section, we have defined an abstract notion of confidential, secure, and authenticated channels. In practice, such channels are realized through cryptographic means. Agents first execute some key establishment protocol in order to generate secret session keys. Then they *encapsulate* the messages supposedly sent over a channel using these session keys. A standard case for secure channels consists in using session keys to encrypt subsequent messages. How to encrypt the message is defined by the *encapsulation*. In Section 3.1, we provide a generic definition of encapsulations and identify properties needed for encapsulations to allow for authentication, confidential, and secure channels. We continue in Section 3.2 by characterizing the composition of a key establishment protocol with a process using abstract channels.

3.1 Encapsulation

For our composition result, we *tag* encapsulations and processes. These tags are used to distinguish the parts of a message that correspond to encapsulations from the ones coming from processes. Formally, a tag is a constant from $\mathcal{F}_{cst}$, hence known to the attacker. Given a set $\mathsf{Tag} \subseteq \mathcal{F}_{cst}$, we say that a term t is a Tag-term when for all $t' \in st(t)$, if $t' = \mathsf{f}(t_1, \ldots, t_n)$ for some $\mathsf{f} \in \mathcal{F}_c \backslash \{\langle\,\rangle\}$ and some terms $t_1, \ldots, t_n$ then $t_1 = \langle \mathsf{a}, u \rangle$ for some term u and $\mathsf{a} \in \mathsf{Tag}$.

▶ **Definition 5.** A Tag-encapsulation is a pair $(\mathcal{E}, \mathsf{F})$ where $\mathcal{E}$ is a Tag-term of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\mathsf{F} \subseteq \mathcal{T}(\mathcal{F}_{key}, \mathcal{X})$ such that $vars(\mathcal{E}) = \{x, x_1, \ldots, x_n\}$, $\{\mathcal{E}, x_1, \ldots, x_n\} \vdash x$ and for all $t \in st(\mathcal{E})$,

- if $t = \mathsf{f}(v)$ with $\mathsf{f} \in \mathcal{F}_{key}$ then $v \in \{x_1, \ldots, x_n\} \cup \mathcal{F}_{cst}$
- if $t = \mathsf{f}(w, t_1, \ldots, t_n)$ and there exists a f-decomposition rule with $\mathsf{f}(x, u_1, \ldots, u_n), v_1, \ldots, v_m$ as premises then for all $j \in \{1, \ldots, m\}$, for all $i \in \{1, \ldots, n\}$, $v_j = \mathsf{g}(y)$ and $y \in vars(u_i)$ implies $t_i \in \{x_1, \ldots, x_n\} \cup \mathcal{F}_{cst}$. Intuitively, if a f-decomposition rule may be applied to a subterm of an encapsulation using a non atomic key $\mathsf{g}(t_i)$ then t_i must be a variable or a constant.

We denote x by $\mathsf{t}_{\mathcal{E}}$ and $(x_1, \ldots, x_n)$ by $\mathsf{X}_{\mathcal{E}}$. Given two encapsulations $(\mathcal{E}, \mathsf{F})$ and $(\mathcal{E}', \mathsf{F}')$, we write $\mathcal{E} \sim \mathcal{E}'$ when there exists a renaming ρ such that $\mathcal{E}\rho = \mathcal{E}'$, $\mathsf{F}\rho = \mathsf{F}'$, $\mathsf{t}_{\mathcal{E}}\rho = \mathsf{t}_{\mathcal{E}'}$ and $\mathsf{X}_{\mathcal{E}}\rho = \mathsf{X}_{\mathcal{E}'}$. We denote by $\mathcal{E}(t, t_1, \ldots, t_n)$ the term obtained from $\mathcal{E}$ by substituting x by t and x_i by t_i.

In an encapsulation $(\mathcal{E}, \mathsf{F})$, the variable $\mathsf{t}_{\mathcal{E}}$ will be instantiated by the message sent on the channel implemented by the encapsulation whereas the variables in $\mathsf{X}_{\mathcal{E}}$ will be instantiated by the session keys. Note that $\{\mathcal{E}, x_1, \ldots, x_n\} \vdash x$ indicates that an encapsulated messages may always be retrieved using the session keys. The terms in F represent the public keys that can be used to deduce the term encapsulated or to generate an encapsulation with a new message without revealing the session keys.

▶ **Example 6.** In Example 4, we described how the session keys $\mathsf{f}_1(k_{\mathsf{R}}, k_{\mathsf{P}})$ and $\mathsf{f}_2(k_{\mathsf{R}}, k_{\mathsf{P}})$ are established in the BAC protocol. The ICAO standard states that in any other protocol executed after BAC, the messages exchanged should be of the form $\langle u, \mathsf{mac}(\langle \mathsf{b}, u \rangle, \mathsf{f}_1(k_{\mathsf{R}}, k_{\mathsf{P}}))\rangle$ with $u = \mathsf{senc}(\langle \mathsf{b}, M \rangle, \mathsf{f}_2(k_{\mathsf{R}}, k_{\mathsf{P}}))$ for some data M and tag b. This represents in fact the encapsulation of M with the session keys $\mathsf{f}_1(k_{\mathsf{R}}, k_{\mathsf{P}})$ and $\mathsf{f}_2(k_{\mathsf{R}}, k_{\mathsf{P}})$. In our formalism, the encapsulation is defined as $(\mathcal{E}_{\mathsf{BAC}}, \emptyset)$ where $\mathcal{E}_{\mathsf{BAC}} = \langle t, \mathsf{mac}(\langle \mathsf{b}, t \rangle, x_2)\rangle$ with $t = \mathsf{senc}(\langle \mathsf{b}, x \rangle, x_1)$, $\mathsf{t}_{\mathcal{E}_{\mathsf{BAC}}} = x$ and $\mathsf{X}_{\mathcal{E}_{\mathsf{BAC}}} = (x_1, x_2)$.

We use tags to distinguish the encapsulations from the messages actually sent over the network. However, a process can implement different types of channels using different encapsulations with the same tags. We need to ensure that the security of an encapsulation is not compromised when used with other encapsulations. Therefore, to state the different properties that encapsulations must satisfy, we consider a set of encapsulations and not only a unique one. These conditions are easily met by standard encapsulations.

▶ **Definition 7.** Let $\mathcal{S}_e = \mathcal{S}_a \uplus \mathcal{S}_c \uplus \mathcal{S}_s$ be a set of Tag-encapsulations. We say that $\mathcal{S}_e$ allows authentic, confidential and secure channels if the following properties are satisfied: Let $(\mathcal{E}_1, \mathsf{F}_1), \ldots, (\mathcal{E}_n, \mathsf{F}_n) \in \mathcal{S}_e$. Assume that the variables in $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are disjoint. Let σ be a ground substitution such that $dom(\sigma) = vars(\mathcal{E}_1, \ldots, \mathcal{E}_n)$ and let Φ be a ground frame such that $\mathsf{Tag} \cap st(\sigma, \Phi) = \emptyset$. Let I be the set of $i \in \{1, \ldots, n\}$ such that $\Phi \cdot [\mathcal{E}_k\sigma]_{k=1}^n \vdash \mathsf{t}_{\mathcal{E}_i}\sigma$.

1. For all $i \in \{1, \ldots, n\}$, $\forall u \in \mathcal{T}(\mathcal{F}_{key}, \mathsf{X}_{\mathcal{E}_i}\sigma)$, if $\Phi \cdot [\mathcal{E}_k\sigma]_{k=1}^n \vdash u$ then $\Phi \cdot [\mathsf{t}_{\mathcal{E}_k}\sigma]_{k\in I} \vdash u$.
2. For all $i, i' \in \{1, \ldots, n\}$, $\forall u \in st(\mathcal{E}_i) \setminus \mathcal{X}$, $\forall v \in st(\mathcal{E}_{i'}) \setminus \mathcal{X}$, if u and v are unifiable and $\mathsf{root}(u) \neq \{\langle\,\rangle\}$ then $img(mgu(u, v)) \subset \mathcal{X}$.

Moreover, an encapulation is *authentic*, that is $(\mathcal{E}_i, \mathsf{F}_i) \in \mathcal{S}_a$ if it satisfies the properties **[Can read]** and **[Cannot write]**. An encapulation is *confidential*, that is $(\mathcal{E}_i, \mathsf{F}_i) \in \mathcal{S}_c$ if it satisfies the properties **[Cannot read]** and **[Can write]**. Finally, an encapulation is *secure*, that is $(\mathcal{E}_i, \mathsf{F}_i) \in \mathcal{S}_c$ if it satisfies the properties **[Cannot read]** and **[Cannot write]**.

For all ground substitution σ' such that $\mathsf{Tag} \cap st(\sigma') = \emptyset$, if we denote $J = I - i$ then

3. **[Can read]** $[\mathcal{E}_i] \cdot \mathsf{F}_i \vdash \mathsf{t}_{\mathcal{E}_i}$
4. **[Cannot read]** $\Phi \cdot [\mathcal{E}_k\sigma]_{k=1}^n \vdash \mathsf{t}_{\mathcal{E}_i}\sigma$ implies $\Phi \cdot [\mathsf{t}_{\mathcal{E}_k}\sigma]_{k\in J} \vdash \mathsf{t}_{\mathcal{E}_i}\sigma \vee \exists x \in \mathsf{X}_{\mathcal{E}_i}.\Phi \cdot [\mathsf{t}_{\mathcal{E}_k}\sigma]_{k\in J} \vdash x\sigma$
5. **[Can write]** $\Phi \cdot [\mathcal{E}_k\sigma]_{k=1}^n \vdash \mathcal{E}_i\sigma' \Leftrightarrow \varphi \vee \big(\Phi \cdot [\mathsf{t}_{\mathcal{E}_k}\sigma]_{k\in I} \vdash \mathsf{t}_{\mathcal{E}_i}\sigma' \wedge \Phi \cdot [\mathsf{t}_{\mathcal{E}_k}\sigma]_{k\in I} \vdash \mathsf{F}_i\sigma'\big)$
6. **[Cannot write]** $\Phi \cdot [\mathcal{E}_k\sigma]_{k=1}^n \vdash \mathcal{E}_i\sigma'$ implies either φ or the following property:

$$\exists x \in \mathsf{X}_{\mathcal{E}_i}.\Phi' \vdash x\sigma' \wedge \big((\exists j \in N.\mathsf{t}_{\mathcal{E}_i}\sigma' = \mathsf{t}_{\mathcal{E}_j}\sigma \wedge \mathsf{X}_{\mathcal{E}_i}\sigma' \cap \mathsf{X}_{\mathcal{E}_j}\sigma \neq \emptyset) \vee \Phi' \vdash \mathsf{t}_{\mathcal{E}_i}\sigma')\big)$$

where $\varphi = \exists j \in N.(\mathcal{E}_i \sim \mathcal{E}_j \wedge \mathcal{E}_i\sigma' = \mathcal{E}_j\sigma)$, $N = \{1, \ldots, n\}$ and $\Phi' = \Phi \cdot [\mathsf{t}_{\mathcal{E}_k}\sigma]_{k\in I}$.

The set $\mathcal{S}_a$ (resp. $\mathcal{S}_c$, $\mathcal{S}_s$) represents the sets of encapsulations that can be used to implement authentic (resp. confidential, secure) channels. Property 1 indicates that the session keys or their associated public keys cannot be retrieved directly from an encapsulation. Different encapsulations may use for instance the same encryption scheme. However, Property 2 prevents a part of an encapsulation to be mistaken as session key for another encapsulation. Properties 3 to 6 model the access control of an encapsulation. In particular, the term $\mathsf{t}_{\mathcal{E}}$ of an encapsulation allowing reading access can be derived from the encapsulation $\mathcal{E}$ and its public keys F (Property 3). On the other hand, the term $\mathsf{t}_{\mathcal{E}}$ of an encapsulation not allowing reading access should not be derived from the encapsulation without knowing the session keys $\mathsf{X}_{\mathcal{E}}$ (Property 4). Property 5 indicates that an encapsulation allowing writing access can be deduced only if it was already sent on the network (expressed by formula φ) or by generating it from its public keys F and the term $\mathsf{t}_{\mathcal{E}}$ encapsulated. Lastly, Property 6 models that an encapsulation not allowing writing access cannot be generated by an attacker unless already given or some session keys in $\mathsf{X}_{\mathcal{E}}$ are known. In the latter, Property 6 also states that when the term $\mathsf{t}_{\mathcal{E}}$ is not known to the attacker then he must have extracted it from encapsulations previously received. Most common encapsulations satisfy these properties.

▶ **Theorem 8.** *The following encapsulations are:*
authentic: $\mathcal{E}_{\mathsf{sign}} = \mathsf{sign}(\langle \mathsf{a}_{\mathcal{E}\mathsf{sign}}, x\rangle, x_1)$ *and* $\mathcal{E}_{\mathsf{mac}} = \langle x, \mathsf{h}(\langle \mathsf{a}_{\mathcal{E}\mathsf{mac}}, x, x_1\rangle)\rangle$*;*
confidential: $\mathcal{E}_{\mathsf{aenc}} = \mathsf{aenc}(\langle \mathsf{a}_{\mathcal{E}\mathsf{aenc}}, x\rangle, \mathsf{pk}(x_1))$*;*
secure: $\mathcal{E}_{\mathsf{TLS}} = \mathsf{senc}(\langle \mathsf{a}_{\mathcal{E}\mathsf{TLS}}, x\rangle, x_1)$, $\mathcal{E}_{\mathsf{BAC}} = \langle t, \mathsf{mac}(\langle \mathsf{a}_{\mathcal{E}\mathsf{BAC}}, t\rangle, x_2)\rangle$ *with* $t = \mathsf{senc}(\langle \mathsf{a}_{\mathcal{E}\mathsf{BAC}}, x\rangle, x_1)$, *and* $\mathcal{E}_{\mathsf{signcrypt}} = \mathsf{sign}(\langle \mathsf{a}_{\mathcal{E}\mathsf{signcrypt}}, \mathsf{aenc}(\langle \mathsf{a}_{\mathcal{E}\mathsf{signcrypt}}, x\rangle, \mathsf{pk}(x_1))\rangle, x_2)$.

where $\mathsf{a}_{\mathcal{E}\mathsf{sign}}, \mathsf{a}_{\mathcal{E}\mathsf{mac}}, \mathsf{a}_{\mathcal{E}\mathsf{aenc}}, \mathsf{a}_{\mathcal{E}\mathsf{TLS}}, \mathsf{a}_{\mathcal{E}\mathsf{BAC}}, \mathsf{a}_{\mathcal{E}\mathsf{signcrypt}}$ *are constants.*

Furthermore, the set of encapsulations $\{(\mathcal{E}_{\mathsf{sign}}, \{\mathsf{vk}(x_1)\}),\ (\mathcal{E}_{\mathsf{mac}}, \emptyset),\ (\mathcal{E}_{\mathsf{BAC}}, \emptyset),\ (\mathcal{E}_{\mathsf{TLS}}, \emptyset),\ (\mathcal{E}_{\mathsf{signcrypt}}, \emptyset),\ (\mathcal{E}_{\mathsf{aenc}}, \{\mathsf{pk}(x_1)\})\}$ *allows for authentic, confidential and secure channels.*

In the rest of this paper, we assume the existence of a set of encapsulations $\mathcal{S}_e$ allowing authentic, secure and confidential channels.

3.2 Composition of protocols

Encapsulations use session keys, which are established by a key exchange protocol. To express the requested property of this protocol, we need to annotate it with events that specify which keys are established for which channels and agents.

Considering a context of channel and agent declarations C and a set of channels S, we denote by $C|_{\overline{S}}$ the context C where all $\mathsf{new}_{ta}\ c$ with $c \in S$ are removed. We denote by $T_{\mathcal{A}gt}$ the set of tuples of agents. We consider special events $\mathsf{Ev} = \{\mathsf{ev}_1, \mathsf{ev}_2, \ldots \in \mathcal{E}v\}$.

▶ **Definition 9.** Let $P = C[R_1, \ldots, R_n]$ be a process with C an agent and channel declaration context such that $R_1, \ldots, R_n$ are roles of agents $A_1, \ldots, A_n$ respectively. Let S be a set of channels such that $channels(C) \cap S = \emptyset$. Let ρ be a mapping from S to $T_{\mathcal{A}gt} \times \mathcal{S}_e$. We say that a process $\tilde{P}$ is an annotation of P under ρ if $\tilde{P} = C[R'_1, \ldots, R'_n]$ where for all $i \in \{1, \ldots, n\}$,

$$R'_i = R_i.\mathsf{event}_{A_i}(\mathsf{ev}_i(c_1, ta_1, ts_1, tp_1)).\ldots.\mathsf{event}_{A_i}(\mathsf{ev}_i(c_m, ta_m, ts_m, tp_m))$$

where $\{c_1, \ldots, c_m\} = \{c \in dom(\rho) \mid c\rho = (ta, (\mathcal{E}, \mathsf{F})) \wedge A_i \in st(ta)\}$ and $\forall j \in \{1, \ldots, m\}$, $c_j\rho = (ta_j, (\mathcal{E}, \mathsf{F}))$, $ts_j = (u_1, \ldots, u_{|\mathsf{X}_\mathcal{E}|})$, $tp = \mathsf{F}(u_1, \ldots, u_{|\mathsf{X}_\mathcal{E}|})$ for some $(\mathcal{E}, \mathsf{F})$ and terms $u_1, \ldots, u_{|\mathsf{X}_\mathcal{E}|}$ such that if $c \in \mathcal{C}h_a$ (resp. $\mathcal{C}h_c$, $\mathcal{C}h_s$) then $(\mathcal{E}, \mathsf{F})$ allows authentic (resp. confidential, secure) channels.

At the end of each role R_i, we add the events ev_i for the channels $c_1, \ldots, c_m$ that the agent is supposed to establish. Events $\mathsf{ev}_i(c, ta, ts, tp)$ are composed of four elements: a channel c that the agent wants to instantiate, a tuple of agents ta indicating who is sharing the channel c, a tuple of session keys ts that will be used in the encapsulation $(\mathcal{E}, \mathsf{F})$ to implement c, and lastly a tuple tp of public keys associated to the session keys and F. Typically, we will require that the session keys in ts remain secret for honest agents while their associated public keys in F are made public.

▶ **Example 10.** Continuing Example 4 and thanks to Theorem 8, the encapsulation $(\mathcal{E}_{\mathsf{BAC}}, \emptyset)$ provides the passport and reader with a secure channel, denoted $c_s \in \mathcal{C}h_s$, once BAC has been executed. The fact that BAC is supposed to establish a secure channel for P and R is expressed by the mapping $\rho = \{c_s \to ((\mathsf{P}, \mathsf{R}), (\mathcal{E}, \emptyset))\}$. The corresponding annotation of BAC under ρ is as follows:

$$\begin{aligned} \tilde{BAC} = \quad & C_{BAC}[R_\mathsf{P}.\mathsf{event}_\mathsf{P}(\mathsf{ev}_1(c_s, (\mathsf{P}, \mathsf{R}), (\mathsf{f}_1(y, k_\mathsf{P}), \mathsf{f}_2(y, k_\mathsf{P})))) \\ & \mid R_\mathsf{R}.\mathsf{event}_\mathsf{R}(\mathsf{ev}_2(c_s, (\mathsf{P}, \mathsf{R}), (\mathsf{f}_1(k_\mathsf{R}, w), \mathsf{f}_2(k_\mathsf{R}, w))))] \end{aligned}$$

where $C_{BAC}[\_] = \mathsf{ag}(\mathsf{R}, \{\mathsf{R}\}, \emptyset, \emptyset).!\mathsf{ag}(\mathsf{P}, \mathcal{P}, \emptyset, \{ke[\mathsf{P}], km[\mathsf{P}], data[\mathsf{P}]\}).\_$. Note that the session keys are different and reflect the respective views on the session keys of the passport and the reader.

▶ **Definition 11.** Let C and C' be two channel and agent declaration contexts. We say that C and C' are composable if there exist contexts C_1, C_2, C'_1, C'_2 such that C_1 and C'_1 are sequences of agent declarations with $ba(C_1) \cap ba(C'_1) = \emptyset$, $C = C_1[C_2]$, $C' = C'_1[C'_2]$ and C_2, C'_2 only differ from the content of $\mathcal{K}_{pub}$, $\mathcal{K}_{prv}$ in the instances of $\mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub}, \mathcal{K}_{prv})$.

We define their composition, denoted $\mathsf{C}^{C,C'}$, as the context $C_1[C_1'[C_3]]$ with C_3 being the context C_2 where all instances of $\mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub}, \mathcal{K}_{prv})$ are replaced by $\mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub} \cup \mathcal{K}_{pub}', \mathcal{K}_{prv} \cup \mathcal{K}_{prv}')$ and $\mathsf{ag}(A, \mathcal{A}, \mathcal{K}_{pub}', \mathcal{K}_{prv}')$ is in C_2'.

The composability of the channel and agent declaration contexts ensures that the roles of the process Q can be sequentially composed with the roles of the process P. For instance, they should have similar replications, agent declarations or even channel declarations. However, we do not require that an agent in P and Q to have the same private ($\mathcal{K}_{prv}$) or public ($\mathcal{K}_{pub}$) data. We also allow an agent to be declared in one context but not in the other one if declared upfront.

▶ **Example 12.** One of the protocols that are executed after BAC is the Passive Authentication protocol which provides an authentication mechanism proving that the content of the RFID chip is authentic. In fact the ICAO standard also indicates that the chip must contain a signature by the Document Signer authority (D) of a hash of the private data $data[\mathsf{P}]$, $sod \stackrel{def}{=} \mathsf{sign}(\langle \mathsf{a}, \mathsf{h}(\langle \mathsf{a}, data[\mathsf{P}]\rangle)\rangle, sk[\mathsf{D}])$. During the Passive Authentication protocol, after receiving on the secure channel a challenge from the reader, the passport sends back this signature that is checked by the reader.

$\mathsf{R} \rightarrow_{sec} \mathsf{P} : \mathsf{read}$
$\mathsf{P} \rightarrow_{sec} \mathsf{R} : \langle data, \mathsf{sign}(\langle \mathsf{a}, \mathsf{h}(\langle \mathsf{a}, data\rangle)\rangle, sk)$

where sk is the signing key of the Document Signer authority. In our calculus, the roles of the reader (Q_{R}) and of the passport (Q_{P}) can be described as follows:

$Q_{\mathsf{P}} = \mathsf{in}_{\mathsf{P}}(c_s, \mathsf{read}).\mathsf{out}_{\mathsf{P}}(c_s, \langle data[\mathsf{P}], sod\rangle)$
$Q_{\mathsf{R}} = \mathsf{out}_{\mathsf{R}}(c_s, \mathsf{read}).\mathsf{in}_{\mathsf{R}}(c_s, \langle x', \mathsf{sign}(\langle \mathsf{a}, \mathsf{h}(\langle \mathsf{a}, x'\rangle)\rangle, sk[\mathsf{D}])\rangle)$

The complete representation of the system is given by $PA = C_{PA}[\mathsf{new}_{(R,P)}\ c_s.(Q_P \mid Q_R)]$ where C_{PA} is the following context:

$$C_{PA} = \mathsf{ag}(\mathsf{D}, \{\mathsf{D}\}, \{\mathsf{vk}(sk[\mathsf{D}])\}, \{sk[\mathsf{D}]\}).\mathsf{ag}(\mathsf{R}, \{\mathsf{R}\}, \emptyset, \emptyset).!\mathsf{ag}(\mathsf{P}, \mathcal{P}, \emptyset, \{data[\mathsf{P}]\}).\_$$

Continuing Example 10, C_{PA} and C_{BAC} are composable and $\mathsf{C}^{C_{PA},C_{BAC}}$ is the context:

$$\mathsf{ag}(\mathsf{D}, \{\mathsf{D}\}, \{\mathsf{vk}(sk[\mathsf{D}])\}, \{sk[\mathsf{D}]\}).\mathsf{ag}(\mathsf{R}, \{\mathsf{R}\}, \emptyset, \emptyset).!\mathsf{ag}(\mathsf{P}, \mathcal{P}, \emptyset, \{ke[\mathsf{P}], km[\mathsf{P}], data[\mathsf{P}]\}).\_$$

Let S be a set of channels. Let ρ be a mapping from S to $T_{\mathcal{A}gt} \times \mathcal{S}_e$. We say that two processes P and Q are *composable under* ρ if $P = C[R_1, \ldots, R_n]$, $Q = C'[R_1', \ldots, R_n']$ where R_i, R_i' are roles of the same agent A_i for $i = 1 \ldots n$, C and $C'|_{\overline{S}}$ are composable and for all $c \in dom(\rho)$, if $c\rho = (ta, (\mathcal{E}, \mathsf{F}))$ then for all $i \in \{1, \ldots, n\}$, $c \in ch_{A_i}(Q)$ is equivalent to $A_i \in ta$. This reflects the fact that agents using channel c should be explicitly listed as authorized agents for c.

The composability between P and Q ensures that the agents in Q sharing abstract authentic, confidential and secure channels are correctly represented in ρ.

▶ **Definition 13.** Let S be a set of channels. Let ρ be a mapping from S to $T_{\mathcal{A}gt} \times \mathcal{S}_e$. Let $P = C[R_1, \ldots, R_n]$ and $Q = C'[R_1', \ldots, R_n']$ two closed composable processes under ρ.

For all $\tilde{P} = C[\tilde{R}_1, \ldots, \tilde{R}_n]$ annotations of P under ρ, the implementation of Q by $\tilde{P}$ through ρ, denoted $\tilde{P} \cdot^{\rho} Q$, is the process $C_0[R_1.R_1'', \ldots, R_n.R_n'']$ where $C_0 = \mathsf{C}^{C,C'|_{\overline{S}}}$ and for all $i \in \{1, \ldots, n\}$, R_i'' is defined as R_i' where all instances of $\mathsf{out}_A(c, u)$ (resp. $\mathsf{in}_A(c, u)$) are replaced by $\mathsf{out}_A(c_{pub}, \mathcal{E}\sigma)$ (resp. $\mathsf{in}_A(c_{pub}, \mathcal{E}\sigma)$) when $c\rho = (ta, (\mathcal{E}, \mathsf{F}))$, $\mathsf{t}_{\mathcal{E}}\sigma = u$ and $\mathsf{event}_A(\mathsf{ev}_i(c, ta, \mathsf{X}_{\mathcal{E}}\sigma, \mathsf{F}\sigma))$ is in $\tilde{R}_i$ for some substitution σ.

▶ **Example 14.** Continuing Example 12, the implementation of PA by $B\tilde{A}C$ through ρ is thus the process $B\tilde{A}C \cdot^{\rho} PA = \mathsf{C}^{C_{PA},C_{BAC}}[R_P.Q_P' \mid R_R.Q_R']$ where Q_P' and Q_R' are defined

as follows:

$$Q'_{\mathsf{P}} = \mathsf{in}_{\mathsf{P}}(c_{pub}, \mathcal{E}_{\mathsf{BAC}}(\mathsf{read}, K_1, K_2)).\mathsf{out}_{\mathsf{P}}(c_{pub}, \mathcal{E}_{\mathsf{BAC}}(\langle data[\mathsf{P}], sod\rangle, K_1, K_2))$$
$$Q'_{\mathsf{R}} = \mathsf{out}_{\mathsf{R}}(c_{pub}, \mathcal{E}_{\mathsf{BAC}}(\mathsf{read}, K'_1, K'_2)).\mathsf{in}_{\mathsf{R}}(c_{pub}, \mathcal{E}_{\mathsf{BAC}}(\langle x, \mathsf{sign}(\langle \mathsf{a}, \mathsf{h}(\langle \mathsf{a}, x\rangle)\rangle, sk[\mathsf{D}])\rangle, K'_1, K'_2))$$

with $K_1 = \mathsf{f}_1(y, k_{\mathsf{P}})$, $K_2 = \mathsf{f}_2(y, k_{\mathsf{P}})$, $K'_1 = \mathsf{f}_1(k_{\mathsf{R}}, w)$, $K'_2 = \mathsf{f}_2(k_{\mathsf{P}}, w)$. Note that the ICAO standard describes in fact the Passive Authentication protocol as the process $C[Q'_P \mid Q'_R]$ (without tags). With our result, we may study the simpler process $C[\mathsf{new}_{(P,R)}\ c_s.(Q_P \mid Q_R)]$.

4 Security property

It is easy to state secrecy in our formalism, using a special event $\mathsf{Sec} \in \mathcal{E}v$: any term occurring in a Sec event should remain secret unless the corresponding session involves a dishonest agent.

▶ **Definition 15.** Let Q be closed process containing contains some events of the form $\mathsf{Sec}(t, (A_1, \ldots, A_n))$ where t is a term and $A_1, \ldots, A_n$ are some agents. Let Φ be a closed frame. We say that Q preserves secrecy if for all $(Q, \emptyset, \emptyset, \emptyset) \xRightarrow{ev_1 \cdot \ldots \cdot ev_m} (Q', \Phi', \mu', \theta')$, for all $i \in \{1, \ldots, n\}$, if $ev_i = \mathsf{Sec}(t', (A'_1, \ldots, A'_n))$ for some t' and some honest agents $A'_1, \ldots, A'_n$ then $\Phi' \not\vdash t'$.

We may also specify the properties requested from a key exchange protocol P: P should preserve the secrecy of the session keys occurring in its events and should ensure that the associated public keys are public. Moreover, P also needs to ensure that a session key cannot be used to implement two different channels and that honest agents sharing a channel will share the same session keys for this channel. In such a case, we say that P is a *secure channel establishment protocol*.

▶ **Definition 16.** Let $P = C[R_1, \ldots, R_n]$ be a closed process. Let $\tilde{P}$ be an annotation of P under some mapping ρ. We say that $\tilde{P}$ is a *secure channel establishment protocol* when for all $(\tilde{P}, \emptyset, \emptyset, \emptyset) \xRightarrow{e_1 \cdot \ldots \cdot e_m} (P', \Phi', \mu', \theta')$, for all $i \in \{1, \ldots, m\}$, if $e_i = ev(c, ta, (s_1, \ldots, s_\ell), (u_1, \ldots, u_q))$ such that $ev \in \mathsf{Ev}$, all agents in ta are honest then for all $k \in \{1, \ldots, \ell\}$, $\Phi' \not\vdash s_k$ and for all $k \in \{1, \ldots, q\}$, $\Phi' \vdash u_k$. Moreover, for all $j \in \{1, \ldots, m\}$, if $ev_j = ev'(c', ta', (s'_1, \ldots, s'_{\ell'}), (u'_1, \ldots, u'_{q'}))$ for some $ev' \in \mathsf{Ev}$, some channel c', some tuple ta' of agents and some tuples $(s'_1, \ldots, s'_{\ell'})$ and $(u'_1, \ldots, u'_{q'})$ of terms then

- either $ta \neq ta'$ or $c \neq c'$ or $ev = ev'$ implies $\forall k \in \{1, \ldots, \ell\}, \forall k' \in \{1, \ldots, \ell'\}$, $s_k \neq s'_{k'}$
- or one of the two following properties is satisfied :
  - $(s_1, \ldots, s_\ell) = (s'_1, \ldots, s'_{\ell'})$ and $(u_1, \ldots, u_q) = (u'_1, \ldots, u'_{q'})$.
  - $\forall k \in \{1, \ldots, \ell\}, \forall k' \in \{1, \ldots, \ell'\}$, $s_k \neq s'_{k'}$.

The first item indicates that the session keys used for a channel between some honest agents are necessarily different from session keys used for a different channel between any kind of agents, whether they are honest, dishonest or a mix of both. The second item requires that for matching channels and sets of agents, either the session keys perfectly match or they are all different.

We are now ready to state our main result: if P is a secure channel establishment protocol and if Q preserves secrecy using some secure, confidential, or authentic channels, then Q may safely use P to implement its channels. The proof of Theorem 17 is available in a companion report [5].

▶ **Theorem 17.** *Let* tag_A *and* tag_B *be two disjoint sets of tags. Let* $\mathcal{S}_e$ *be a set of* tag_A*-encapsulation allowing authentic, confidential, and secure channels. Let* ρ *be a mapping from channels to* $T_{\mathcal{A}gt} \times \mathcal{S}_e$*. Let* P *and* Q *be two closed executable composable* tag_B*-processes under* ρ *such that* P *and* Q *do not share names and* $fa(P) = fa(Q) = \emptyset$*. Let* $\tilde{P}$ *be an annotation of* P *under* ρ*. If* $\tilde{P}$ *is secure and* Q *preserves secrecy then* $\tilde{P} \cdot^{\rho} Q$ *preserves secrecy as well.*

For simplicity, we prove secure composition w.r.t. secrecy properties but we believe that our result could be easily extended to trace properties.

Sketch of proof. The proof first relies on that fact that the reachability properties are preserved by disjoint parallel composition. In particular, the process $\tilde{P} \mid Q$ is a secure channel establishment protocol and preserves secrecy. The rest of the proof consists in showing that any trace of $\tilde{P} \cdot^{\rho} Q$ is also a trace of $\tilde{P} \mid Q$ with a frame that induces a similar attacker knowledge. More specifically, properties from Definition 7 ensure that tag_B-terms generated by the attacker or obtained from the encapsulations in $\tilde{P} \cdot^{\rho} Q$ do not give any relevant knowledge to the attacker and can be replaced by fresh names. This allows us to obtain a trace without tag_B-terms and so without encapsulations. Lastly, since $\tilde{P} \mid Q$ is a secure channel establishment protocol, we can always match two encapsulations having same session keys with the corresponding abstract channel in $\tilde{P} \mid Q$. ◀

▶ **Example 18.** Continuing Example 14, the annotation under ρ of the Basic Access Control $\tilde{BAC}$ is secure and the Passive Authentication $C_{PA}[\mathsf{new}\ c_s.(Q_P.\mathsf{event}_{\mathsf{P}}(\mathsf{Sec}(data[\mathsf{P}], (\mathsf{P}, \mathsf{R}))) \mid Q_R)]$ preserves secrecy (of the private data). Hence, thanks to Theorems 8 and 17, the implementation of PA by $\tilde{BAC}$ through ρ, $\mathsf{C}^{C_{PA}, C_{BAC}}[R_P.Q'_P.\mathsf{event}_{\mathsf{P}}(\mathsf{Sec}(data[\mathsf{P}], (\mathsf{P}, \mathsf{R}))) \mid R_R.Q'_R]$, preserves secrecy.

5 Case studies

We show that our approach can be applied to deployed protocols such as the biometric passport or TLS applied to 3D-secure. As an application, we show that the automatic analysis through the CL-Atse tool can be significantly speed up when the number of sessions goes higher.

5.1 Biometric passport

Our running example is the combination of the Basic Access Control (BAC) protocol with the Passive Authentication (PA) protocol from the electronic passports. Actually, PA is not the only protocol executed after BAC. Another authentication mechanism is used to prevent cloning of the passport chip. This protocol, called *Active Authentication protocol* (AA), also uses the same session keys and encapsulations than PA. Using the CL-Atse tool [18], we show for different scenarios that BAC is a secure channel establishment protocol and that PA and AA both preserve secrecy. Thanks to our main result, this yields security of the combined protocol, where BAC implements the secure channel of PA and AA. For comparaison purpose, we also analyze directly the combined protocol with CL-Atse. These analysis are reported in Section 5.3

5.2 TLS and 3D-secure

Our results also apply to other complex systems. We study the *Visa 3D-secure protocol* [17] used by several websites for internet banking and that relies on secure channels implemented

by the well known TLS protocol. The Visa 3D secure protocol is an authenticated payment method between a card holder and a merchant during an electronic payment. This protocol aims to ensure authentication of the card holder as well as confirmation that the card holder is authorized by his bank to make the payment. Lastly, the protocol also aims to ensure the secrecy of the card holder's banking information, the payment amount and other data.

The protocol involves four types of participants: a card holder (C), a merchant (M), a centralized structure called Visa Directory Servers (DS) and the card issuer's servers called Access Control Servers (ACS). The main role of the Visa Directory Servers is to transfer card holder's information between the Access Control Servers and the merchant.In itself, the 3D secure protocol is already a complex protocol with multiple exchanges of messages. But the protocol also requires most messages to be exchanged trough a TLS channel. More specifically, messages of the 3D secure protocol shall be encrypted with a symmetric session key previously established with TLS. In our model, this means that the messages are encapsulated by $(\mathcal{E}_{\mathsf{TLS}}, \emptyset)$, as defined in Theorem 8.

The well known TLS protocol [15, 9] aims at establishing a secure channel between a client and a server. Using the CL-Atse tool, we show that TLS (Basic TLS handshake, in the RSA mode) is indeed a secure channel establishment protocol.

Note that for one session of the Visa 3D secure protocol yields four sessions of the TLS protocol: one channel between C and M, between C and ACS, between ACS and DS and finally between M and DS. This renders the verification of even one session of 3D secure protocol with the channels implemented by TLS a complex task (more than thirty five messages exchanged per session).

5.3 Analysis with CL-Atse

We applied the automatic verification tool CL-Atse [18] on a Dell T1700 computer (16 Go RAM, 3.40 GHz CPU). The corresponding time of analysis are displayed below.

		Computation time (in seconds, timeout set to 24 hours)							
protocols		TLS & 3D secure		BAC & PA		BAC & AA		BAC & PA & AA	
complete system (C) or separated analysis (S)		S	C	S	C	S	C	S	C
number of sessions considered	1	0.2	0.1	0.7	0.1	0.7	0.1	0.7	0.2
	2	1350	time out	6.2	1.6	6.2	1.6	6.5	43156
	3	time out	time out	9133	time out	9133	time out	9185	time out

Amongst the tools able to verify security protocols for a bounded number of sessions, CL-Atse is well known and considered to be one of the fastest. However, in the case of the 3D-secure protocol, the tool already fails to verify one session with all channels implemented as we reached a time out set to 24 hours of computation. Thus, to obtain meaningful results with the 3D-secure protocol, we considered the case where only the channel between the card holder and the merchant is implemented. Already in this case, we can see a clear benefit from analyzing separately 3D-secure and TLS when considering two sessions. Indeed, the verification can be performed under 25 minutes when analysing the protocols separately whereas the tool was reaching a time out when considering the complete system. We obtain similar results with the Basic Access Control protocol, the Active Authentication protocol and the Passive Authentication protocol. Note that for verification tools handling unbounded number of sessions (*e.g.* ProVerif [3], Tamarin [14]), the gain in time would probably be less significant since these tools do not systematically explore all interleavings.

6 Conclusion

We have shown how to securely compose a protocol with the implementation of its channels. We have provided a characterization for the three most common types of channels: secure, confidential, and authentic channels. We plan to consider other types of communication channels like anonymous channels. This will certainly require to extend our approach to equivalence properties.

Our composition result holds for a class of primitives that encompasses all standard cryptographic primitives. We plan to extend it to a larger class of primitives, including in particular exclusive or or homomorphic encryption.

Our result assumes a light tagging of the primitives, to ensure that an encapsulation cannot be confused with a message coming from the protocols. While tagging is reasonable, it is not often done in practice. On the other hand standard protocols typically enjoy some non unifiability properties that prevent such confusion. We believe that our result could be extended to a general notion of non unifiability of the terms, without having to require explicit tagging.

Acknowledgments. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865, project ProSecure.

References

1 Machine readable travel document. Technical Report 9303, International Civil Aviation Organization, 2008.

2 M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

3 Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. CSFW'01*, 2001.

4 Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: Tagging enforces termination. In Andrew Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, April 2003.

5 V. Cheval, V. Cortier, and E. Le-Morvan. Secure refinements of communication channels. Research report RR-8790, Inria, 2015.

6 Ştefan Ciobâcă and Véronique Cortier. Protocol composition for arbitrary primitives. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 322–336, Edinburgh, Scotland, UK, July 2010. IEEE Computer Society Press.

7 Véronique Cortier and Stéphanie Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, February 2009.

8 Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol composition logic (PCL). *Electr. Notes Theoretical Computer Science*, 172:311–358, 2007.

9 T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2 (rfc 5246). Technical report, IETF, 2008.

10 Thomas Gibson-Robinson, Allaa Kamil, and Gavin Lowe. Verifying layered security protocols. *Journal of Computer Security*, 23(3), 2015.

11 Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3–4):409–433, 2004.

12 Joshua D. Guttman. Establishing and preserving protocol security goals. *Journal of Computer Security*, 22(2):203–267, 2004.

13 Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.

14 Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification, 25th International Conference, CAV 2013, Princeton, USA, Proc.*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.

15 Christopher Meyer and Jorg Schwenk. Lessons learned from previous ssl/tls attacks : A brief chronology of attacks and weaknesses. In *IACR Cryptology ePrint*, 2013.

16 Sebastian Moedersheim and Luca Viganò. Sufficient conditions for vertical composition of security protocols. In *ASIACCS*, pages 435–446, 2014.

17 Vijayakrishnan Pasupathinathan, Josef Pieprzyk, Huaxiong Wang, and Joo Yeon Cho. Formal analysis of card-based payment systems in mobile services. In *Fourth Australian information security workshop, conferences in research and practise in information security*, pages 213–220, 2006.

18 Mathieu Turuani. The CL-Atse Protocol Analyser. In *Term Rewriting and Applications – Proc. of RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286, Seattle, WA, USA, 2006.