Types for Security Protocols [1]

Riccardo FOCARDI [a] and Matteo MAFFEI [b]
[a] *University of Venice, Italy*
[b] *Saarland University, Germany*

Abstract. We revise existing type-based analyses of security protocols by devising a core type system for secrecy, integrity and authentication in the setting of spi-calculus processes. These fundamental security properties are usually studied independently. Our exercise of considering all of them in a uniform framework is interesting under different perspectives: (i) it provides a general overview of how type theory can be applied to reason on security protocols; (ii) it illustrates and compares the main results and techniques in literature; (iii) perhaps more importantly, it shows that by combining techniques deployed for different properties, existing type-systems can be significantly simplified.

1. Introduction

Predicting the behaviour of a protocol or program by just inspecting its code is a very intriguing challenge which can be approached in different ways. Techniques such as abstract interpretation [18] or control flow analysis [33,12] aim at defining sound abstractions of the actual semantics which overapproximate the behaviour of the protocol: all the concrete executions are guaranteed to be captured by the abstract semantics. This allows for developing efficient analyses which can certify the correctness of a protocol with respect to some target (security) property P: if the abstraction satisfies P we are guaranteed that all the concrete runs will also satisfy P.

Type theory takes a somehow complementary perspective. Saying, for example, that a message has a certain type allows for statically check that such a message will be used in a controlled way so to avoid violating the target property P. Instead of abstracting the behaviour of the protocol and check P over the abstraction, we devise a set of rules that dictate how typed data and protocols should be programmed so to respect P. One interesting aspect of this approach is that it forces understanding *why* and *how* security is achieved. This is particularly useful for security protocols whose flaws often derive by some degree of ambiguity in the role that messages play in achieving a certain goal. Type-based reasoning is therefore particularly valuable, as it forces one to clarify protocol specifications by making explicit the underlying security mechanisms.

A toy example. We consider *Confidentiality*, i.e., the property of data being accessible only by authorized users. In a (idealized) distributed setting we might think of honest principals sharing secure communication channels. Thus, a simple way to achieve con-

[1] Work partially supported by the initiative for excellence and the Emmy Noether program of the German federal government and by Miur'07 Project SOFT: "*Security Oriented Formal Techniques*".

fidentiality might be to impose that high-confidential (or secret) data are only sent on secure channels. If $\overline{c}\langle M\rangle$ denotes the output of M over channel c we might formalize the above idea as:

$$M : \mathsf{Secret}, c : \mathsf{Secure} \vdash \overline{c}\langle M\rangle$$

meaning that under the hypothesis M is of type Secret and c is of type Secure we can type-check the output $\overline{c}\langle M\rangle$. Of course we do not want to write a different rule for every different case. For example it is clearly safe to even send a public message over a secure channel. For this reason it is useful to introduce a notion of derived type: $\Gamma \vdash M : \mathsf{Secret}$ meaning that M can be given type Secret starting from the type bindings listed in Γ. For example, clearly, $M : \mathsf{Secret}, c : \mathsf{Secure} \vdash M : \mathsf{Secret}$. But we could also say that $M : \mathsf{Public}, c : \mathsf{Secure} \vdash M : \mathsf{Secret}$, since a public data can be safely regarded as secret. The typing rule for the output can be now rewritten as:

$$\frac{\Gamma \vdash M : \mathsf{Secret} \qquad \Gamma \vdash c : \mathsf{Secure}}{\Gamma \vdash \overline{c}\langle M\rangle}$$

The fact that a type T is less demanding than another one T' is usually formalized through a *subtyping* relation $T \leq T'$. Thus, in our case it is enough to state that $\mathsf{Public} \leq \mathsf{Secret}$ in order to allow all public data to be regarded also as secrets. Assuming this, the above typing rule does not actually say much: on a secure channel we can send whatever kind of data, both public and secret. It is instead more interesting to regulate what can be done on insecure channels

$$\frac{\Gamma \vdash M : \mathsf{Public} \qquad \Gamma \vdash c : \mathsf{Insecure}}{\Gamma \vdash \overline{c}\langle M\rangle}$$

Given that $\mathsf{Secret} \not\leq \mathsf{Public}$ we intuitively have that secret data will never be sent on insecure channels.

To be useful, types must be preserved at run-time, i.e., a typed protocol should remain typed when it is executed and, in particular, when variables get bound to names received from the network. In our small toy example, this amounts to specify what type we expect when we receive messages from the network: in particular we are required to give type Secret to all messages coming from secure channels. Noting $c(x).P$ a protocol reading a message M from channel c and binding x with M in the sequent P, we can write:

$$\frac{\Gamma, x : \mathsf{Secret} \vdash P \qquad \Gamma \vdash c : \mathsf{Secure}}{\Gamma \vdash c(x).P}$$

If the channel is secure, x is given type Secret and the sequent P is typed under this assumption, meaning that P must treat x as it were a secret.

A fourth rule might state that when reading from insecure channels we can safely give type public to x. At this point, however, one is tempted to try to find a more succinct way to express these four cases. One attempt might be to equate Secret with Secure and Public with Insecure, but this would lead to an insecure channel being regarded as secure, obviously breaking confidentiality. A more appropriate way to find a connection between data and channel types is to look at their security level. We can write $\mathcal{L}(\mathsf{Secret}) = \mathcal{L}(\mathsf{Secure}) = H$ and $\mathcal{L}(\mathsf{Public}) = \mathcal{L}(\mathsf{Insecure}) = L$, meaning that secret data and secure channels have a high security level, while public data and inse-

cure channels have a low one. With this idea in mind the whole toy type system can be summarized as follows:

$$\frac{\mathcal{L}(T_d) = \mathcal{L}(T_c) \quad \Gamma \vdash M : T_d \quad \Gamma \vdash c : T_c}{\Gamma \vdash \overline{c}\langle M \rangle} \qquad \frac{\mathcal{L}(T_d) = \mathcal{L}(T_c) \quad \Gamma, x : T_d \vdash P \quad \Gamma \vdash c : T_c}{\Gamma \vdash c(x).P}$$

where T_d ranges over Secret and Public and T_c ranges over Secure and Insecure. Intuitively, a message M can be sent over c if its security level does not exceed the one of the channel (a secret message can never be sent on an insecure channel); notice that a public message can be risen to secret via subtyping, and be sent on a secure channel; dually, a message x received from c must be assumed to be at least at the security level of the channel (a message received from a secure channel must be regarded as secret). For example if Γ is c : Insecure, d : Secure we can type-check protocol $c(x).\overline{d}\langle x \rangle$, which forwards messages read from an insecure channel to a secure one. The typing derivation is as follows:

$$\dfrac{\dfrac{\dfrac{\mathsf{Public} \leq \mathsf{Secret}}{\Gamma, x : \mathsf{Public} \vdash x : \mathsf{Secret}} \quad \Gamma, x : \mathsf{Public} \vdash d : \mathsf{Secure}}{\Gamma, x : \mathsf{Public} \vdash \overline{d}\langle x \rangle}}{\Gamma \vdash c(x).\overline{d}\langle x \rangle}$$

If we swap the channels, of course, the protocol becomes not typable as messages read from secure channels should never be forwarded on insecure ones. Formally, $\Gamma \not\vdash d(x).\overline{c}\langle x \rangle$ since $\Gamma, x : \mathsf{Secret} \not\vdash \overline{c}\langle x \rangle$, being c insecure.

Type-based analysis of Security Protocols: an overview. Type-based analysis of security protocols dates back to Abadi's seminal work [1] on *secrecy by typing*. This work focuses on security protocols based on symmetric-key cryptography and on the secrecy of data. The idea is to model cryptographic protocols in the spi-calculus [6] and to verify confidentiality with a type system. One of the fundamental contributions is the methodology used to type-check the opponent: processes manipulating only messages of type Public are shown to be always well-typed (*opponent typability*). This technique allows for type-checking the opponent without posing any constraints on his behavior. The confidentiality property established by the type system is expressed in terms of noninterference: an opponent will never be able to distinguish two protocol executions in which the initial value of sensitive data may differ. Abadi and Blanchet subsequently extended the type system to reason about security protocols based on asymmetric cryptography [4,5]

Among the foundational contributions of this research line, we point out the technique used to type-check encrypted messages. Keys are given a type of the form $\mathsf{Key}^{\ell}[T]$, which dictates the type T of the messages encrypted with that key. The security level ℓ specifies whether the key is possibly known to the opponent (ℓ=Public) or not (ℓ=Secret). The receiver of a ciphertext can thus determine the type of the decrypted message by the type of the key. If the key has type $\mathsf{Key}^{\mathsf{Secret}}[T]$, then the ciphertext comes from a well-typed process and the decrypted message has type T; if the key has type $\mathsf{Key}^{\mathsf{Public}}[T]$, then the ciphertext might come from a well-typed process as well as from the opponent and the continuation process has to be type-checked twice, once with the message being of type T and once with the message being of type Public.

Gordon and Jeffrey proposed a type and effect system for verifying authenticity in cryptographic protocols based on symmetric [23] and asymmetric cryptography [24]. The fundamental idea is to formulate authenticity properties in terms of correspondence assertions [38] and to use dependent types in order to characterize the assertions valid for each message. Correspondence assertions are protocol annotations of the form begin(M) and end(M), marking the begin and the end of a protocol session for authenticating message M. Intuitively, a protocol guarantees authenticity if every end is preceded by a corresponding begin [28]. The type system was subsequently extended to handle conditional secrecy (a refinement of secrecy, where a message is unknown to the adversary unless particular messages or principals are compromised) [25] and protocols based on time-stamps [26].

Bugliesi et al. proposed an alternative technique for the verification of authenticity in security protocols [13,30,15]. This framework is based on a dynamic type and effect system, which exploits a set of tags that uniquely identify the type and effect of encrypted messages. The analysis enjoys strong compositionality guarantees and is well-suited to reason about multi-protocol systems [29], although it assumes a tagging discipline for messages. We refer the interested reader to [14] for a formal comparison between this type and effect system and the above one by Gordon and Jeffrey.

Building upon this research line, Fournet et al. proposed a type system for the verification of authorization policies in security protocols [20]. The idea is to decorate the protocol with assumptions and assertions of the form assume C and assert C, respectively, where C is a logical formula. A protocol is safe if every assertion is entailed by the active assumptions. Authorization policies allow for reasoning about authenticity as well as other security requirements, for instance access control policies. Authorization policies, however, do not capture the freshness of authentication requests and the type system does not handle nonce handshakes. The authors subsequently extended the type system to reason about distributed systems where some of the principals are compromised [21]. Backes et al. further refined the expressivity of the type system to reason about protocols based on zero-knowledge proofs [9].

Even if security protocols are properly designed, security flaws may still affect implementations. For this reason, the analysis of executable code is crucial to provide end-to-end security guarantees. Bengtson et al. [10] recently proposed a framework for the type-based analysis of authorization policies in F# implementations of security protocols. The type system is based on refinement types, which describe the type of values as well as logical formulas that characterize such values. The language is a lambda-calculus with primitives for concurrency, which is used to define the syntax of a large fragment of F# by encoding. One important contribution of this work is the definition of a library of symbolic cryptography in the lambda-calculus. In contrast to previous approaches, cryptographic primitives are not modelled by ad-hoc language primitives and verified by specific typing rules. They are instead defined by symbolic libraries based on *sealing* [32,36,35] and verified by the standard typing rules for functions. This makes the type system easily extensible to a large number of cryptographic primitives.

Outline of this work. We devise a core type system for confidentiality, integrity and authentication starting from pi-calculus processes, in which security is guaranteed via ideal (restricted) channels. This simplified setting allows us to introduce important concepts and basic types for secrecy and integrity, disregarding all the subtleties introduced by cryptographic operations. We then consider a rather rich dialect of spi-calculus with

$M, N, K ::=$	terms	$P, Q, R, O ::=$	processes
x, y, z	variable	$\overline{N}\langle \tilde{M} \rangle.P$	output
a, b, c, d, k, m, n, s	name	$N(\tilde{x}).P$	input
		$\mathbf{0}$	stop
		$P \mid Q$	parallel
		$!P$	replication
		$(\nu a : T)\ P$	restriction
		if $M = N$ then P else Q	conditional

Table 1. Core calculus: terms and processes syntax

symmetric/asymmetric cryptography and digital signature. We show how the types for cryptographic keys can be defined as an extension of the channel types of the pi-calculus: the type transported by a secure channel can be seen as the type of the message encrypted with a secure key. Finally, we add a system of effects to track linearity of nonce usage in challenge-response protocols. Interestingly, the final type-system is much simpler than the ones in literature (e.g., [15,23,24]). We feel that this is mainly due to the benefit of combining in a uniform setting techniques deployed for different properties.

In our study we mix techniques and concepts from the literature, with the main aim of defining a general setting where different contributions can be illustrated and understood. In doing so, we have also borrowed concepts from the language-based security literature (see, e.g., [34] for a survey), in particular for what concerns the dual treatment of confidentiality and integrity. It is worth mentioning that recent language-based secure literature has extended imperative languages with cryptography allowing for the modelling of cryptographic protocols in a language setting (see, e.g., [7,16,17,22,27,37]). It is thus natural to try to bridge the language-based and the process-calculi settings and take advantage from both of them. In summary, our work provides a general overview of how type theory can be applied to reason on security protocols illustrating the main results and techniques in literature; interestingly, it shows that existing type-systems can be significantly simplified by combining techniques originally deployed for verifying different properties.

Note: Due to space constraints, we include in this chapter only the most interesting proofs. A full version of this work is available at [19].

2. Secure Communication in the Pi-Calculus

We introduce a core calculus for reasoning about communication protocols without cryptography. It is essentially a polyadic pi-calculus [31] with a typing annotation for restricted names which will be useful to reason about security and has no semantic import. In fact, to simplify the notation, types will be omitted when unimportant. This calculus allows us to introduce in the simplest possible setting many important concepts, properties and proof techniques. In section 3, we will extend it with various cryptographic primitives and we will show how these primitives can be statically checked so to provide security.

Structural Equivalence

$P \equiv P$
$P \equiv Q \Rightarrow Q \equiv P$
$P \equiv Q\ ,\ Q \equiv R \Rightarrow P \equiv R$
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$
$P \equiv Q \Rightarrow\ !P \equiv !Q$
$P \equiv Q \Rightarrow (\nu a : T)\ P \equiv (\nu a : T)\ Q$

$(\nu a : T)\ \mathbf{0} \equiv \mathbf{0}$
$P \mid \mathbf{0} \equiv P$
$P \mid Q \equiv Q \mid P$
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
$!P \equiv P \mid !P$
$(\nu a : T)\ (\nu b : T')\ P \equiv (\nu b : T')\ (\nu a : T)\ P \quad \text{if } a \neq b$
$(\nu a : T)\ (P \mid Q) \equiv P \mid (\nu a : T)\ Q \text{ if } a \notin \mathit{fn}(P)$

Reduction

$$\overline{N}\langle \tilde{M} \rangle.P \mid N(\tilde{x}).Q \rightarrow P \mid Q\{\tilde{M}/\tilde{x}\} \qquad \text{(RED I/0)}$$
$$\text{if } M = M \text{ then } P \text{ else } Q \rightarrow P \qquad \text{(RED COND 1)}$$
$$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad \text{if } M \neq N \qquad \text{(RED COND 2)}$$

(RED STRUCT)
$$\frac{P' \equiv P \qquad P \rightarrow Q \qquad Q \equiv Q'}{P' \rightarrow Q'}$$

(RED RES)
$$\frac{P \rightarrow Q}{(\nu a : T)\ P \rightarrow (\nu a : T)\ Q}$$

(RED PAR)
$$\frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R}$$

Table 2. Structural Equivalence and Reduction

Syntax. The syntax of the calculus is given in Table 1. For the sake of readability, we let $\tilde{M}$ denote a sequence $M_1, \ldots, M_m$ of terms and $\{\tilde{M}/\tilde{x}\}$ the substitution $\{M_1/x_1\} \ldots \{M_m/x_m\}$. Intuitively, process $\overline{N}\langle \tilde{M} \rangle.P$ outputs the tuple of messages $\tilde{M}$ on channel N and then behaves as P; $N(\tilde{x}).P$ receives a tuple of messages $\tilde{M}$ (where the arity of $\tilde{M}$ and $\tilde{x}$ is the same) from channel N and then behaves as $P\{\tilde{M}/\tilde{x}\}$; $\mathbf{0}$ is stuck; the parallel composition $P \mid Q$ executes P and Q concurrently; the replication $!P$ behaves as an unbounded number of copies of P in parallel; $(\nu a : T)\ P$ generates a fresh name a (of type T) and then behaves as P; finally, $\text{if } M = N \text{ then } P \text{ else } Q$ behaves as P if M is equal to N or as Q otherwise. We will often omit the trailing $\mathbf{0}$ writing, e.g., $\overline{N}\langle \tilde{M} \rangle$ in place of $\overline{N}\langle \tilde{M} \rangle.\mathbf{0}$.

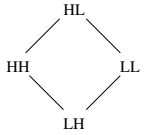
We let $\mathit{fnfv}(M)$ and $\mathit{fnfv}(P)$ denote the free names and variables in term M and process P, respectively. The notion of free names and variables is defined as expected: all names and variables occurring in a term are free; the restriction $(\nu \tilde{a} : \tilde{T})\ P$ is a binder for a with scope P and the input $N(\tilde{x}).P$ is a binder for $\tilde{x}$ with scope P. We implicitly identify processes up to renaming of bound names and variables.

Semantics. The semantics of the calculus is formalized in Table 2 in terms of a *structural equivalence* relation $\equiv$ and a *reduction* relation $\rightarrow$. Structural equivalence $\equiv$ is defined as the least relation satisfying the rules reported in the first part of Table 2. It is an equivalence relation closed with respect to parallel composition, replication and restriction, which essentially allows us to rearrange parallel compositions and restrictions in order to bring processes that should interact close to each other, to unfold replications, and to remove useless restrictions. Reduction $\rightarrow$ is defined as the least relation on closed processes satisfying the rules in the second part of Table 2. Communication is synchronous: the output $\overline{N}\langle \tilde{M} \rangle.P$ synchronizes with an input $N(\tilde{x}).Q$ on the same channel and then reduces to $P \mid Q\{\tilde{M}/\tilde{x}\}$ (rule RED I/0). The equality test $\text{if } M = N \text{ then } P \text{ else } Q$ reduces to P if M is equal to N or to Q otherwise (rules RED COND 1 and 2). Moreover, reduction relation preserves $\equiv$ (RED STRUCT) and is closed with respect to restriction

(RED RES) and parallel composition (RED PAR). In the following, we let $P \rightarrow^* Q$ hold true if P reduces in one or more steps to Q or if P is structurally equivalent to Q.

2.1. Security Levels and Security Properties

In the literature on language-based security, it is common to study confidentiality and integrity together (see, e.g., [34]). Usually, the security level is a pair $\ell_C \ell_I$ specifying, separately, the confidentiality and integrity levels. We consider two possible levels: *High* (H) and *Low* (L). For example, HH denotes a high confidentiality and high integrity value, while LH a public (low confidentiality) and high integrity one. Intuitively, high confidentiality values should never be read by opponents while high integrity values should not be modified by opponents, i.e., when we receive high integrity data we expect they originated at some trusted source.



An important difference between confidentiality and integrity levels is that they are contra-variant: while it is safe to consider a public datum as secret, promoting low integrity to high integrity is unsound, as any data from the opponent could erroneously be considered as coming from a trusted entity. Considering instead as low integrity some high integrity data is harmless, as this basically reduces the assumptions we can make on them. More formally, the confidentiality and integrity preorders are such that $L \sqsubseteq_C H$ and $H \sqsubseteq_I L$. We let ℓ_C and ℓ_I range over $\{L, H\}$, while we let ℓ range over the pairs $\ell_C \ell_I$ with $\ell_C^1 \ell_I^1 \sqsubseteq \ell_C^2 \ell_I^2$ iff $\ell_C^1 \sqsubseteq_C \ell_C^2$ and $\ell_I^1 \sqsubseteq_I \ell_I^2$, giving the standard four-point lattice depicted on the right. Intuitively, moving up in the lattice is safe as both secrecy and integrity preorders are respected.

As we mentioned above, our calculus is typed. The four points of the security lattice are our four basic types and they are used for describing generic terms at the specified security level.

Opponents. Processes representing opponents are characterized by type/level LL meaning that they can read from LL and LH while they can modify LL and HL, reflecting the intuition that information may only flow up in the lattice. In particular, opponents can only generate names of type LL, as formalized below:

Definition 1 (Opponent) *A process O is an opponent if all $(\nu a : T)$ occurring in O are such that $T = LL$.*

We will always assume that free names of processes are low confidentiality and low integrity, since they might be known to and originated by the opponent.

Level of types and terms. Later on, we will introduce more sophisticated types giving additional information about how typed terms should be used. For the moment we do not need to give more detail on types except that they always have an associated security level. In particular, we write $\mathcal{L}(T)$ to denote the associated security level. For the four basic types we trivially have $\mathcal{L}(\ell) = \ell$. Given $\mathcal{L}(T) = \ell_C \ell_I$, we also write $\mathcal{L}_{\mathrm{C}}(T)$ and $\mathcal{L}_{\mathrm{I}}(T)$ to respectively extract from T the confidentiality and integrity levels ℓ_C and ℓ_I. Similarly, given a mapping Γ from terms to types, we denote with $\mathcal{L}_\Gamma(M)$ the level in Γ of a certain term M formally defined as:

$$\mathcal{L}_\Gamma(M) = \begin{cases} \mathcal{L}(\Gamma(M)) & \text{whenever } M \in \mathit{dom}(\Gamma) \\ LL & \text{otherwise} \end{cases}$$

As above, $\mathcal{L}_{\mathrm{C},\Gamma}(M)$ and $\mathcal{L}_{\mathrm{I},\Gamma}(M)$ respectively denote the confidentiality and integrity level associated to M in Γ.

Secrecy. As mentioned above, secrecy refers to the impossibility for an opponent to access/read some data d. This property can be interpreted in two different ways, the latter strictly stronger than the former: (i) the opponent should not learn the exact value of d or (ii) the opponent should not deduce any information about d. We give a small example to illustrate: process $\overline{a}\langle d\rangle.\mathbf{0}$ clearly violates both notions as d is just sent on the network, while process $\mathsf{if}\ d = d'\ \mathsf{then}\ \overline{a}\langle n\rangle.\mathbf{0}$ clearly violates (ii) as *some* information about d is actually leaked, in particular its equality with d', but (i) might still hold; for example if d' is also secret then no intruder will be able to compute d from the output n. Property (ii) is usually called *noninterference*. For lack of space we will only focus on notion (i).

Our definition of secrecy is in the style of [2]. A process P preserves the secrecy of a high confidentiality name d if d cannot be computed by any opponent interacting with P. Notice that, whenever the opponent computes d, he can also send it on a unrestricted (public) channel. Of course the opponent should not know the secret in advance, for this reason we only focus on secrets which are restricted names. For the sake of readability we write the definition for channels of arity 1 (the extension to arity n is immediate).

Definition 2 (Secrecy) *P preserves secrecy if, for all opponents O, whenever $P \mid O \rightarrow^* (\nu d : T)\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle d\rangle.P'')$ we have $\mathcal{L}_{\mathrm{C}}(T) \sqsubseteq_C \mathcal{L}_{\mathrm{C},\Gamma}(b)$, with $\Gamma = d : T, \tilde{a} : \tilde{T}$.*

Notice that Γ is always guaranteed to be a function thanks to implicit alpha renaming of bound names. Intuitively, a process preserves secrecy if its names are always transmitted on channels with at least the same confidentiality level, even when interacting with an arbitrary opponent O. In particular, if d is secret also the channel b is required to be secret, meaning it is one of the names in $\tilde{a}$. Recall, in fact, that $\mathcal{L}_{\mathrm{C},\Gamma}(b)$ returns L for all names not mapped by Γ. Thus, if P preserves secrecy then its secret names will never be transmitted on unrestricted (public) channels.

As an example, process $(\nu d : HL)\ \overline{b}\langle d\rangle$ clearly breaks the secrecy of d by sending it on the unrestricted channel b. Formally, $\mathcal{L}_{\mathrm{C}}(HL) = H \not\sqsubseteq_C L = \mathcal{L}_{\mathrm{C},\Gamma}(b)$, with $\Gamma = d : HL$. Process $(\nu b : HL)\ (\nu d : HL)\ \overline{b}\langle d\rangle$, instead, preserves secrecy.

Integrity. Formalizations of integrity in process calculi literature are, to the best of our knowledge, not so common. We believe, however, that it is convenient to introduce a formal definition of integrity at this point, as it will allow us to dually treat secrecy and integrity guarantees provided by channels and cryptographic keys.

To formalize integrity we need to introduce the type $\mathsf{C}^\ell[\tilde{T}]$ for channels at level ℓ transporting data of type $\tilde{T}$. Since in the core calculus communication is symmetric, the only interesting levels for channels are HH and LL, respectively representing trusted and untrusted channels. We thus limit ℓ to those two levels and we let $\mathcal{L}(\mathsf{C}^\ell[\tilde{T}]) = \ell$.

The notion of integrity is essentially dual to the one of secrecy: we require that any data transmitted on a trusted channel in a position corresponding to high integrity data will always be a high integrity name, i.e., a name originated from some trusted process. Notice, to this purpose, that we have forbidden opponents to generate high integrity names. The formal definition follows:

$$\textsc{Empty}\quad \emptyset \vdash \diamond \qquad\qquad \textsc{Env}\quad \frac{\Gamma \vdash \diamond \qquad M \notin \mathit{dom}(\Gamma) \qquad T = \mathsf{C}^{\ell}[\tilde{T}] \text{ implies } \ell = HH}{\Gamma, M : T \vdash \diamond}$$

Table 3. Core calculus: well-formedness of Γ.

Definition 3 (Integrity) *P preserves integrity if, for all opponents O, whenever $P \mid O \rightarrow^{*} (\nu b : \mathsf{C}^{HH}[T'])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle d\rangle.P'')$ we have $\mathcal{L}_{\mathrm{I},\Gamma}(d) \sqsubseteq_I \mathcal{L}_{\mathrm{I}}(T')$, with $\Gamma = b : \mathsf{C}^{HH}[T'], \tilde{a} : \tilde{T}$.*

We now give an example of a process which breaks the above property.

Example 1 Consider process $(\nu b : \mathsf{C}^{HH}[LH])\ (c(x).\overline{b}\langle x\rangle \mid b(y).P)$. Intuitively, this process reads from the untrusted channel c a value x and forwards it on the trusted channel b. Since x can be low integrity, this process violates the above property. Take, for example, the opponent $\overline{c}\langle d\rangle$ and the reduction:

$$\begin{aligned}
&(\nu b : \mathsf{C}^{HH}[LH])\ (c(x).\overline{b}\langle x\rangle \mid b(y).P) \mid \overline{c}\langle d\rangle \\
&\quad \equiv (\nu b : \mathsf{C}^{HH}[LH])\ (c(x).\overline{b}\langle x\rangle \mid b(y).P \mid \overline{c}\langle d\rangle) \\
&\quad \rightarrow (\nu b : \mathsf{C}^{HH}[LH])\ (\overline{b}\langle d\rangle \mid b(y).P \mid \mathbf{0}) \\
&\quad \rightarrow (\nu b : \mathsf{C}^{HH}[LH])\ (\mathbf{0} \mid P\{d/x\} \mid \mathbf{0})
\end{aligned}$$

In the last step, d is sent on channel b, which should only be used for transmitting high integrity values, but d is not restricted as it comes from the opponent: integrity does not hold. Formally, this can be seen in process $(\nu b : \mathsf{C}^{HH}[LH])\ (\overline{b}\langle d\rangle \mid b(y).P \mid \mathbf{0})$ which transmits over b an unrestricted name d. We have, $\mathcal{L}_{\mathrm{I},\Gamma}(d) = L \not\sqsubseteq_I H = \mathcal{L}_{\mathrm{I}}(LH)$, with $\Gamma = b : \mathsf{C}^{HH}[LH]$.

2.2. A Core Type System

In this section we present a type system to statically enforce secrecy and integrity in the pi-calculus.

Types and Environment. Our types are just levels (of the four point lattice) and channel types, which we introduced above. Formally, type syntax is as follows:

$$T ::= \ell \mid \mathsf{C}^{\ell}[\tilde{T}] \tag{1}$$

where ℓ is the type of data at such a level, and $\mathsf{C}^{\ell}[\tilde{T}]$ is the type of channels at level ℓ transporting data of type $\tilde{T}$. The typing environment Γ is a set of bindings between names/variables and their respective type T. The well formedness of Γ is defined by the typing rules in Table 3. We require that Γ does not contain multiple bindings for the same value. Additionally, we only admit in Γ (trusted) channels at level HH. As expected, $\mathcal{L}(\ell) = \ell$ and $\mathcal{L}(\mathsf{C}^{\ell}[\tilde{T}]) = \ell$.

Typing Terms. Types for terms are formalized in Table 4: they are the ones in Γ plus the ones derivable by *subtyping*. Intuitively, the subtyping relation $T \leq T'$ specifies when a value of type T can be used in place of a value of type T', thus making the type system more permissive. Formally, $\leq$ is defined as the least preorder such that:

$$\frac{\Gamma \vdash \diamond \qquad M : T \text{ in } \Gamma}{\Gamma \vdash M : T}\ \textsc{Atom} \qquad\qquad \frac{\Gamma \vdash M : T' \qquad T' \leq T}{\Gamma \vdash M : T}\ \textsc{Subsumption}$$

Table 4. Core calculus: typing of terms.

$$\begin{array}{l} \ell_1 \leq \ell_2 \text{ whenever } \ell_1 \sqsubseteq \ell_2 \\ LL \leq \mathsf{C}^{LL}[LL, \ldots, LL] \\ \mathsf{C}^{\ell}[\tilde{T}] \leq \ell \end{array} \tag{2}$$

The first condition means that rising the data security level is harmless. $LL \leq \mathsf{C}^{LL}[LL, \ldots, LL]$ means that any untrusted data can be used as an untrusted channel to transmit untrusted data. Since we forbid LL channels in Γ, this is the only way an untrusted channel can be typed. $\mathsf{C}^{\ell}[\tilde{T}] \leq \ell$ means that channels can be considered as generic data at their security level ℓ. For trusted channels this can never be reversed: once a trusted channel is considered as a datum, it can never be used again as a channel. In fact, $HH \not\leq LL$. Notice that, as expected, subtyping respects the security level lattice. Formally:

Remark 1 (Level Subtyping) $T \leq T'$ implies $\mathcal{L}(T) \sqsubseteq \mathcal{L}(T')$.

Characterizing channel types. We point out some interesting properties for channel types. First, if a term has a channel type of level HH, then this is precisely the type specified in the typing environment, i.e., the channel type has not been derived via subsumption. In fact, the only channel types derivable by subtyping are the ones at level LL.

Proposition 1 (High Channels) $\Gamma \vdash N : \mathsf{C}^{HH}[\tilde{T}]$ *implies* $N : \mathsf{C}^{HH}[\tilde{T}]$ *is in* Γ.

Untrusted LL channels can only be used to transmit untrusted messages. This is a consequence of the fact LL channels cannot be declared in Γ and are only derived via subsumption.

Proposition 2 (Low Channels) $\Gamma \vdash N : \mathsf{C}^{LL}[\tilde{T}]$ *implies* $\tilde{T} = LL, \ldots, LL$.

We also prove that LL and HH are the only admissible security levels for channels, i.e., channel types at level HL and LH are never derivable. This is a consequence of Γ only allowing HH channels and of $\leq$ only deriving LL ones.

Proposition 3 (Channel Levels) $\Gamma \vdash N : \mathsf{C}^{\ell}[\tilde{T}]$ *implies* $\ell \in \{LL, HH\}$.

Finally, a fundamental property of our type system is that the type of a channel of a given arity is unique. This is a consequence of the three above results.

Corollary 1 (Uniqueness of Channel Types) *If* $\Gamma \vdash N : \mathsf{C}^{\ell}[\tilde{T}]$ *and* $\Gamma \vdash N : \mathsf{C}^{\ell'}[\tilde{T'}]$ *with* $|\tilde{T}| = |\tilde{T'}|$ *then* $\mathsf{C}^{\ell}[\tilde{T}] = \mathsf{C}^{\ell'}[\tilde{T'}]$.

The proof of these properties is simple and left as an exercise to the reader.

$$\textsc{Stop}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}} \qquad \textsc{Par}\ \frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \textsc{Repl}\ \frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad \textsc{Res}\ \frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a : T)\, P}$$

$$\textsc{Cond}\ \frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : T' \qquad \Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash \mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q} \qquad \textsc{In}\ \frac{\Gamma, \tilde{x} : \tilde{T} \vdash P \qquad \Gamma \vdash N : \mathsf{C}^{\ell}[\tilde{T}]}{\Gamma \vdash N(\tilde{x}).P}$$

$$\textsc{Out}\ \frac{\Gamma \vdash \tilde{M} : \tilde{T} \qquad \Gamma \vdash P \qquad \Gamma \vdash N : \mathsf{C}^{\ell}[\tilde{T}]}{\Gamma \vdash \overline{N}\langle \tilde{M} \rangle.P}$$

Table 5. Core calculus: typing processes.

Typing Processes. Table 5 dictates how processes should deal with typed channels and values. We use the concise notation $\Gamma \vdash \tilde{M} : \tilde{T}$ for denoting $\forall i \in [1, m]$, $\Gamma \vdash M_i : T_i$. Rules STOP, PAR, REPL, RES and COND simply check that the subprocesses and terms are well-typed under the same Γ, enriched with $a : T$ in case of RES. Intuitively, these rules do not directly impose any restriction. The only interesting rules are, in fact, IN and OUT which respectively state that terms received from and sent to the network are of type $\tilde{T}$, as dictated by the channel type $\mathsf{C}^{\ell}[\tilde{T}]$. Notice that, since input binds the variables $\tilde{x}$, we add $\tilde{x} : \tilde{T}$, i.e., $x_1 : T_1, \ldots, x_m : T_m$, in Γ when typing the sequent process P.

Example 2 Consider again process $(\nu b : \mathsf{C}^{HH}[LH])\ (c(x).\overline{b}\langle x \rangle \mid b(y).P)$ of example 1. We have seen it does not guarantee integrity, as data read from the untrusted channel c are forwarded on the trusted one b. Intuitively, it does not type-check as x is received as LL (i.e., from the environment) and should be lowered to LH in order to be transmitted over b. Recall that we always assume free names such as c to be of type LL, since they can be thought of as under the control of the opponent. We let Γ be $c : LL, b : \mathsf{C}^{HH}[LH]$ and we show that $c(x).\overline{b}\langle x \rangle$ cannot be type-checked under Γ. Notice, in fact, that after type-checking the initial restriction, $b : \mathsf{C}^{HH}[LH]$ is added into Γ. Notice also that, via subsumption, from $\Gamma \vdash c : LL$ and $LL \leq \mathsf{C}^{LL}[LL]$ we obtain $\Gamma \vdash c : \mathsf{C}^{LL}[LL]$. Formally, typing would proceed as follows (read it bottom-up):

$$\textsc{In}\ \frac{\textsc{Out}\ \dfrac{\cancel{\Gamma, x : LL \vdash x : LH} \qquad \Gamma \vdash b : \mathsf{C}^{HH}[LH]}{\Gamma, x : LL \vdash \overline{b}\langle x \rangle} \qquad \Gamma \vdash c : \mathsf{C}^{LL}[LL]}{\Gamma \vdash c(x).\overline{b}\langle x \rangle}$$

The crucial part is that from $x : LL$ we can never prove $x : LH$ since $LL \not\leq LH$. The above example formally shows the importance of considering integrity levels contravariant, as previously discussed: a low-integrity variable can never be considered as high-integrity. Below we will formally proof that typing ensures integrity, thus processes violating integrity, as the above one, can never type-check.

Example 3 Let us consider now a simple protocol where A sends to B a fresh message of level HH on a channel of type $\mathsf{C}^{HH}[HH]$. The typing derivation for the process modelling this protocol is shown below (rule names are omitted for lack of space):

$$\dfrac{\dfrac{\dfrac{\dfrac{c : \mathsf{C}^{HH}[HH], m : HH \vdash \diamond}{c : \mathsf{C}^{HH}[HH], m : HH \vdash \mathbf{0}}}{c : \mathsf{C}^{HH}[HH], m : HH \vdash \overline{c}\langle m\rangle}}{c : \mathsf{C}^{HH}[HH] \vdash (\nu m : HH)\, \overline{c}\langle m\rangle} \qquad \dfrac{\dfrac{c : \mathsf{C}^{HH}[HH], x : HH \vdash \diamond}{c : \mathsf{C}^{HH}[HH], x : HH \vdash \mathbf{0}}}{c : \mathsf{C}^{HH}[HH] \vdash c(x)}}{c : \mathsf{C}^{HH}[HH] \vdash (\nu m : HH)\, \overline{c}\langle m\rangle \mid c(x)}}{\emptyset \vdash (\nu c : \mathsf{C}^{HH}[HH])\, ((\nu m : HH)\, \overline{c}\langle m\rangle \mid c(x))}$$

Notice that the variable x has type HH, so our type system guarantees that what is received by B is both at high confidentiality and high integrity. □

2.3. Properties of the Type System

The next lemmas state some standard properties of our type system. The strengthening lemma states that removing from Γ bindings relative to names not occurring free in the judgment preserve typing. In fact, those names do not contribute in any way to derive the judgment. We let $\mathit{fnfv}(\diamond) = \emptyset$ and $\mathit{fnfv}(M : T) = \mathit{fnfv}(M) = \{M\}$. We write $\Gamma \vdash \mathcal{J}$ to denote the three possible judgments $\Gamma \vdash \diamond$, $\Gamma \vdash M : T$ and $\Gamma \vdash P$.

Lemma 1 (Strengthening) If $\Gamma, M : T \vdash \mathcal{J}$ and $M \notin \mathit{fnfv}(\mathcal{J})$ then $\Gamma \vdash \mathcal{J}$.

The weakening lemma states that extending Γ preserves typing, as long as the extended environment is well-formed. Intuitively, adding new (well-formed) bindings does not compromise typing.

Lemma 2 (Weakening) $\Gamma \vdash \mathcal{J}$ and $\Gamma, M : T \vdash \diamond$ imply $\Gamma, M : T \vdash \mathcal{J}$.

Finally, substituting variables with terms of the appropriate type has no effect on typing.

Lemma 3 (Substitution) If $\Gamma, x : T \vdash \mathcal{J}$ and $\Gamma \vdash M : T$, then $\Gamma \vdash \mathcal{J}\{M/x\}$.

Before proving that the type system enforces both secrecy and integrity, it is important to show that it does not restrict opponent's capabilities. Intuitively, an opponent is untyped as it is not willing to follow any discipline we might want to impose to trusted, typed, processes. However, our theorems are all based on typed processes. It is thus important that the type-system is developed so to avoid any restriction on LL data and channels, so that any opponent can be typed without actually restricting its capabilities. This is what we prove:

Proposition 4 (Opponent typability) *Let O be an opponent and let $\mathit{fn}(O) = \{\tilde{a}\}$. Then $\tilde{a} : LL \vdash O$.*

The proof of these properties is left as an exercise to the interested reader. We now prove the fundamental result underlying type safety: typing is preserved by structural congruence and reduction. Thanks to this result and to the previous proposition, we will be guaranteed that when running a typed process in parallel with a (typed) opponent we will always obtain a typed process. This will allow us to show that secrecy and integrity are preserved at run-time.

Proposition 5 (Subject congruence and reduction) *Let $\Gamma \vdash P$. Then*

1. $P \equiv Q$ *implies* $\Gamma \vdash Q$*;*
2. $P \to Q$ *implies* $\Gamma \vdash Q$.

Proof:
1. In order to deal with the symmetry of $\equiv$ we prove a stronger fact: $P \equiv Q$ or $Q \equiv P$ implies $\Gamma \vdash Q$. We proceed by induction on the derivation of $P \equiv Q$. We need the following easy result:

$$\Gamma \vdash \mathcal{J} \text{ implies } \Gamma \vdash \diamond. \tag{3}$$

We just prove the interesting base cases (and their symmetric counterparts). The remaining ones are all trivial and left as an exercise to the reader.

$(\nu a : T)\ \mathbf{0} \equiv \mathbf{0}$ We have $\Gamma \vdash (\nu a : T)\ \mathbf{0}$. This judgment can only be proved by RES, which implies $\Gamma, a : T \vdash \mathbf{0}$. By (3), $\Gamma, a : T \vdash \diamond$. By Lemma 1 (Strengthening), we obtain $\Gamma \vdash \diamond$. Finally, by STOP, we get $\Gamma \vdash \mathbf{0}$.
The result holds also for symmetric counterpart (i.e., $\mathbf{0} \equiv (\nu a : T)\ \mathbf{0}$), since we can derive $\Gamma, a : T \vdash \diamond$ from $\Gamma \vdash \diamond$ by Lemma 2 (Weakening) and to conclude by STOP and RES.

$(\nu a : T)\ (P \mid Q) \equiv P \mid (\nu a : T)\ Q$ if $a \notin fn(P)$ We know $\Gamma \vdash (\nu a : T)\ (P \mid Q)$, which implies $\Gamma, a : T \vdash P$ and $\Gamma, a : T \vdash Q$. By Lemma 1 (Strengthening), since $a \notin fn(P)$, we get $\Gamma \vdash P$. By RES and PAR, we get $\Gamma \vdash P \mid (\nu a : T)\ Q$.
For the symmetric counterpart $P \mid (\nu a : T)\ Q \equiv (\nu a : T)\ (P \mid Q)$, we have $\Gamma \vdash P$ and $\Gamma, a : T \vdash Q$. By (3), we get $\Gamma, a : T \vdash \diamond$. By Lemma 2 (Weakening), we obtain $\Gamma, a : T \vdash P$. The thesis follows by PAR and RES.

$(\nu a : T)\ (\nu b : T')\ P \equiv (\nu b : T')\ (\nu a : T)\ P\ (a \neq b)$ The judgment $\Gamma \vdash (\nu a : T)\ (\nu b : T')\ P$ can only be proved by RES, which implies $\Gamma, a : T, b : T' \vdash P$ thus $\Gamma, b : T', a : T \vdash P$ since Γ is a set. Notice that the side condition $a \neq b$ is crucial, since we would otherwise have $(\nu a : T)\ (\nu b : T')\ P$ equivalent by α-renaming (if $a \notin \mathit{fn}(P)$) to $(\nu a : T)\ (\nu a : T')\ P\{a/b\} \equiv (\nu a : T')\ (\nu a : T)\ P\{a/b\}$, which is equivalent, again by α-renaming, to $(\nu a : T')\ (\nu b : T)\ P$. This process would not necessarily be well-typed since the type of b has changed.

The remaining (inductive) cases are all trivial. For example, $P \equiv Q\ ,\ Q \equiv R \Rightarrow P \equiv R$ is proved by noticing that $\Gamma \vdash P$ and $P \equiv Q$ imply, by induction, that $\Gamma \vdash Q$. From $Q \equiv R$, again by induction, we get the thesis $\Gamma \vdash R$. The symmetric counterparts of these rules are the same as the original ones except that P and Q are exchanged, so no additional proof is needed.

2. The proof is by induction on the derivation of $P \to Q$. Base cases RED COND 1 and 2 are trivially proved by observing that $\Gamma \vdash \mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ requires $\Gamma \vdash P$ and $\Gamma \vdash Q$. Case RED I/O is proved by observing that Corollary 1 (Uniqueness of Channel Types), IN, and OUT imply $\Gamma, \tilde{x} : \tilde{T} \vdash Q$ and $\Gamma \vdash N : \mathsf{C}^{\ell}[\tilde{T}]$ and $\Gamma \vdash \tilde{M} : \tilde{T}$ and $\Gamma \vdash P$. By applying Lemma 3 (Substitution), we obtain $\Gamma \vdash Q\{\tilde{M}/\tilde{x}\}$ and, by PAR, we get $\Gamma \vdash P \mid Q\{\tilde{M}/\tilde{x}\}$.

The inductive cases are all trivial. We just mention that (RED STRUCT) is based on item 1 of this lemma. □

$M, N, K ::=$ terms		$P, Q, R, O ::=$	processes
$\ldots$	as in Table 1	$\ldots$	as in Table 1
$\mathrm{ek}(K)$	encryption key	case M of $\{\!\lvert\tilde{x}\rvert\!\}^{\mathsf{s}}_{K}$ in P	sym decryption
$\mathrm{vk}(K)$	verification key	case M of $\{\!\lvert\tilde{x}\rvert\!\}^{\mathsf{a}}_{K}$ in P	asym decryption
$\{\!\lvert\tilde{M}\rvert\!\}^{\mathsf{s}}_{K}$	sym encryption	case M of $[\tilde{x}]_K$ in P	signature check
$\{\!\lvert\tilde{M}\rvert\!\}^{\mathsf{a}}_{K}$	asym encryption		
$[\tilde{M}]_K$	digital signature		

Table 6. Syntax for cryptographic messages and cryptographic operations

Secrecy and Integrity by typing. We finally prove that well-typed processes preserve secrecy and integrity.

Theorem 1 (Secrecy and Integrity for $\vdash$) *If $\tilde{n} : LL \vdash \Gamma$, then P preserves both secrecy and integrity.*

Proof:
Let O be an opponent. By Proposition 4 (Opponent typability) we have that $\mathit{fn}(O) = \{\tilde{a}\}$ implies $\tilde{a} : \tilde{LL} \vdash O$. Let $\mathit{fn}(O) \setminus \mathit{dom}(\Gamma) = \{\tilde{b}\}$ be the free names of O not occurring in Γ. Now let $\Gamma' = \Gamma, \tilde{b} : LL$. From $\Gamma \vdash \diamond$, we clearly have that $\Gamma' \vdash \diamond$. By Lemma 2 (Weakening), we have that $\Gamma' \vdash P$ and $\Gamma' \vdash O$. By PAR, we obtain $\Gamma' \vdash P \mid O$. We now have two separate proofs for secrecy and integrity:

Secrecy Let $P \mid O \rightarrow^* (\nu d : T)\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle d\rangle.P'')$. By Proposition 5 (Subject congruence and reduction) we get $\Gamma' \vdash (\nu d : T)\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle d\rangle.P'')$ which implies $\Gamma', d : T, \tilde{a} : \tilde{T} \vdash \overline{b}\langle d\rangle.P''$, by repeatedly applying RES and finally by PAR.

Let $\Gamma'' = \Gamma', d : T, \tilde{a} : \tilde{T}$. By rule OUT we necessarily have that $\Gamma'' \vdash b : \mathsf{C}^{\ell}[T']$ and $\Gamma'' \vdash d : T'$ with $T \leq T'$ and thus $\mathcal{L}_{\mathrm{C}}(T) \sqsubseteq_C \mathcal{L}_{\mathrm{C}}(T')$ by Remark 1 (Level Subtyping).

If $\mathcal{L}_{\mathrm{C}}(T) = L$ we have nothing to prove as we certainly have $\mathcal{L}_{\mathrm{C}}(T) \sqsubseteq_C \mathcal{L}_{\mathrm{C},d:T,\tilde{a}:\tilde{T}}(b)$. Assume then $\mathcal{L}_{\mathrm{C}}(T) = H$. This implies $\mathcal{L}_{\mathrm{C}}(T') = H$. By Proposition 3 (Channel Levels), $\ell \in \{LL, HH\}$, and by Proposition 2 (Low Channels) we are also guaranteed that $\ell = HH$ since, otherwise, T' would necessarily be LL. Then, Proposition 1 (High Channels) proves that $b : \mathsf{C}^{\ell}[T']$ is in Γ'', thus also $\mathcal{L}_{\mathrm{C},d:T,\tilde{a}:\tilde{T}}(b) = H$, giving the thesis.

Integrity Let now $P \mid O \rightarrow^* (\nu b : \mathsf{C}^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle c\rangle.P'')$. By Proposition 5 (Subject congruence and reduction) we get $\Gamma' \vdash (\nu b : \mathsf{C}^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle c\rangle.P'')$, which implies $\Gamma', b : \mathsf{C}^{HH}[T], \tilde{a} : \tilde{T} \vdash \overline{b}\langle c\rangle.P''$, by repeatedly applying RES and finally by PAR.

Let $\Gamma'' = \Gamma', b : \mathsf{C}^{HH}[T], \tilde{a} : \tilde{T}$. Rule OUT requires $\Gamma'' \vdash b : \mathsf{C}^{\ell}[T']$ and $\Gamma'' \vdash c : T'$. By Corollary 1 (Uniqueness of Channel Types), we obtain that $\mathsf{C}^{\ell}[T'] = \mathsf{C}^{HH}[T]$, thus $T = T'$. From $\Gamma'' \vdash c : T$ we necessarily have that $c : T''$ is in Γ'' with $T'' \leq T$ and, by Remark 1 (Level Subtyping), $\mathcal{L}(T'') \sqsubseteq \mathcal{L}(T)$. Notice that $\mathcal{L}_{\mathrm{I}}(T) = H$ implies $\mathcal{L}_{\mathrm{I}}(T'') = H$, as H is the lowest possible level. Since $\mathcal{L}_{\mathrm{I},\mathrm{b}:\mathsf{C}^{HH}[\mathrm{T}'],\tilde{\mathrm{a}}:\tilde{\mathrm{T}}}(c) = \mathcal{L}_{\mathrm{I}}(T'')$ we get the thesis. □

	K^+	K^-
Symmetric encryption	K	K
Asymmetric encryption	$\mathrm{ek}(K)$	K
Digital signature	K	$\mathrm{vk}(K)$

$$\mathsf{case}\ \langle \tilde{M} \rangle_{K^+}\ \mathsf{of}\ \langle \tilde{x} \rangle_{K^-}\ \mathsf{in}\ P \quad \rightarrow \quad P\{\tilde{M}/\tilde{x}\} \quad (\textsc{Dec/Check})$$

Table 7. Semantics of cryptographic operations (extends Table 2)

3. Spi Calculus

Our study on types for cryptographic protocols is developed on a polyadic variant of the spi-calculus [6]. This calculus extends the pi-calculus in order to explicitly reason about protocols based on symmetric encryptions, asymmetric encryptions, and digital signatures.

Syntax and semantics. We extend the syntax of the calculus by introducing (i) terms that represent keys and ciphertexts and (ii) processes that describe cryptographic operations, as shown in Table 6. Term $\mathrm{ek}(K)$ denotes the public encryption key corresponding to the private key K, and term $\mathrm{vk}(K)$ is the public verification key corresponding to the signing key K (hence K has to be kept secret, while $\mathrm{ek}(K)$ and $\mathrm{vk}(K)$ can be published); $\{\!|\tilde{M}|\!\}^{\mathsf{s}}_K$, $\{\!|\tilde{M}|\!\}^{\mathsf{a}}_K$, and $[\tilde{M}]_K$ denote, respectively, the symmetric and asymmetric encryption and the digital signature of the tuple of terms $\tilde{M}$. Notice that we model cryptographic schemes where encryption and verification keys can be recovered from the corresponding decryption and signing keys, respectively. In other words, decryption and signing keys can be seen as key-pairs themselves. We believe this approach provides a more succinct theory but we could model as easily cryptographic schemes where neither key can be retrieved from the other, as done for instance in the original presentation of the spi-calculus [6].

In the following, we write u to denote a name or a variable. It will also be convenient to write $\langle \tilde{M} \rangle_K$ to denote a generic encryption/signature term when its exact nature is unimportant. We will also use the notation K^+ and K^- to respectively denote encryption/signature keys and their decryption/verification counterparts, as specified in Table 7 together with the semantics of cryptographic operations. Intuitively, process $\mathsf{case}\ M\ \mathsf{of}\ \langle \tilde{x} \rangle_{K^-}\ \mathsf{in}\ P$ tries to decrypt or check the signature of M with key K^- and behaves as $P\{\tilde{M}/\tilde{x}\}$ if it succeeds, i.e., when M is $\langle \tilde{M} \rangle_{K^+}$, or gets stuck otherwise.

Example 4 Let us consider the Blanchet authentication protocol [11]. This protocol is modelled in the spi-calculus as follows:

$$\text{Protocol} \triangleq (\nu k_A : T_A)\ (\nu k_B : T_B)\ (\text{Initiator} \mid \text{Responder})$$

For the moment, let us ignore the typing annotations. We first generate two fresh key pairs for A and B, respectively, and then run the processes modelling the initiator B and the responder A in parallel.

$$\begin{aligned}\text{Initiator} \triangleq\ & (\nu k : T_k)\ \overline{c}\langle \{\!|[A, B, k]_{k_B}|\!\}^{\mathsf{a}}_{\mathrm{ek}(k_A)} \rangle . c(x_e) . \\ & \mathsf{case}\ x_e\ \mathsf{of}\ \{\!|x_m|\!\}^{\mathsf{s}}_k\ \mathsf{in}\ \mathbf{0}\end{aligned}$$

The initiator B generates a fresh session key k, signs A and B's identifiers along with k, encrypts this signature with A's encryption key, and outputs the resulting ciphertext on the free channel c, which represents the network. Hence B waits for the response, decrypts it using the session key k, and concludes the protocol session. The process modelling the responder is reported below:

$$\begin{aligned}\text{Responder} \triangleq\ & c(x_e).\mathsf{case}\ x_e\ \mathsf{of}\ \{\!|x_s|\!\}^{\mathsf{a}}_{k_A}\ \ \mathsf{in}\ \ \mathsf{case}\ x_s\ \mathsf{of}\ [x_A, x_B, x_k]_{\mathrm{vk}(k_B)}\ \ \mathsf{in} \\ & \mathsf{if}\ A = x_A\ \mathsf{then}\ (\nu m : HH)\ \overline{c}\langle\{\!|m|\!\}^{\mathsf{s}}_{x_k}\rangle\end{aligned}$$

The responder A receives the challenge, decrypts the ciphertext and verifies the enclosed signature, checks that the first signed message is her own identifier, generates a fresh HH message m and sends it to the initiator encrypted under the received session key x_k. □

4. Types and Integrity (Revised)

Types. We statically characterize the usage of cryptographic keys by extending the syntax of types as follows:

$$\begin{aligned} T &::= \ldots \text{ as in Equation 1} \mid \mu\mathsf{K}^{\ell}[\tilde{T}] \\ \mu &::= \mathsf{Sym} \mid \mathsf{Enc} \mid \mathsf{Dec} \mid \mathsf{Sig} \mid \mathsf{Ver} \end{aligned} \tag{4}$$

These types are close in spirit to the ones for channels. The type $\mu\mathsf{K}^{\ell}[\tilde{T}]$ describes keys at security level ℓ that are used to perform cryptographic operations on terms of type $\tilde{T}$. Depending on the label μ, this type may describe symmetric, encryption, decryption, signing, or verification keys.

Example 5 Let us consider the process illustrated in Example 4. Type T_k of the session key is $\mathsf{SymK}^{HH}[HH]$ as it is trusted ($\ell = HH$), symmetric ($\mu = \mathsf{Sym}$) and transports HH terms. The type T_B of B's signing key k_B is $\mathsf{SigK}^{HH}[LL, LL, T_k]$ and the type of the corresponding (public) verification key $\mathrm{vk}(k_B)$ is $\mathsf{VerK}^{LH}[LL, LL, T_k]$, since this trusted, i.e., high integrity, key-pair is used to sign two LL identifiers and a symmetric session key of type T_k. The type T_A of A's decryption key k_A is $\mathsf{DecK}^{HH}[HH]$ and the type of the corresponding encryption key $\mathrm{ek}(k_A)$ is $\mathsf{EncK}^{LH}[HH]$. This key-pair is indeed used to encrypt a signature which is at high confidentiality, since it contains a secret key, and high integrity, since B has generated it respecting all the types dictated by the signing key. □

Secrecy and Integrity. The definition of secrecy for cryptographic protocols is the same as the one given in Section 2.1, i.e., the opponent should not be able to send high-confidentiality data on public channels. The integrity property, however, has to be revised to take into account the cryptographic setting.

Intuitively, we say that M is a high integrity term if it is either (i) a restricted name bound to a high integrity type, as before, or (ii) a ciphertext or a signature obtained from a secure HH key, in which the integrity of the enclosed messages respects the integrity level dictated by the key type. We achieve this by adapting the definition of $\mathcal{L}_{\mathrm{I},\Gamma}(M)$ as

follows (as before, we focus on ciphertexts of arity 1, since the extension to an arbitrary arity is immediate):

$$\mathcal{L}_{\mathrm{I},\Gamma}(u) = \begin{cases} \mathcal{L}_{\mathrm{I}}(\Gamma(u)) & \text{whenever } u \in \mathit{dom}(\Gamma) \\ L & \text{otherwise} \end{cases}$$

$$\mathcal{L}_{\mathrm{I},\Gamma}(\mathrm{ek}(K)) = \mathcal{L}_{\mathrm{I},\Gamma}(\mathrm{vk}(K)) = \mathcal{L}_{\mathrm{I},\Gamma}(K)$$

$$\mathcal{L}_{\mathrm{I},\Gamma}(\langle M \rangle_{K^+}) = \begin{cases} H & \text{if } \Gamma(K) = \mu\mathsf{K}^{HH}[T] \text{ and} \\ & \quad \mathcal{L}_{\mathrm{I},\Gamma}(M) \sqsubseteq_I \mathcal{L}_{\mathrm{I}}(T) \\ L & \text{otherwise} \end{cases}$$

Intuitively, we want to ensure that high-integrity variables get replaced only by high-integrity terms. This is formalized by the following definition, which extends Definition 3 (Integrity) so to deal with cryptographic operations.

Definition 4 (Integrity with Cryptography) *P preserves integrity if, for all opponents O, whenever*

$$P \mid O \to^* (\nu b : \mathsf{C}^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle M \rangle.P'') \text{ or}$$
$$P \mid O \to^* (\nu k^+ : \mu\mathsf{K}^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \mathsf{case}\ \langle M \rangle_{k^+}\ \mathsf{of}\ \langle x \rangle_{k^-}\ \mathsf{in}\ P'')$$

then $\mathcal{L}_{\mathrm{I},\Gamma}(M) \sqsubseteq_I \mathcal{L}_{\mathrm{I}}(T)$ with $\Gamma = b : \mathsf{C}^{HH}[T], k^+ : \mu\mathsf{K}^{HH}[T], \tilde{a} : \tilde{T}$.

As already noticed, Γ is always guaranteed to be a function thanks to the implicit alpha renaming of bound names. The definition above considers only symmetric ciphertexts and signatures, since K^+ is required to be a name by restriction. Asymmetric ciphertexts in general do not provide integrity guarantees, since they might have been generated by the attacker. In the following, we will see that our type system can in some cases statically determine whether or not a certain ciphertext comes from the attacker (for instance, if the ciphertext was signed with a high-integrity key used to sign self-generated ciphertexts). For the sake of simplicity, however, we preferred to exclude asymmetric ciphertexts from the semantic definition of integrity, since we would otherwise need to explicitly annotate decryptions that are supposed to be applied to high-integrity ciphertexts.

5. A Type System for Cryptographic Protocols

This section extends the type system studied in Section 2.2 to cryptographic protocols. We take the fundamental ingredients of the type system for secrecy proposed by Abadi and Blanchet in [3], showing how these ideas can be smoothly refined to also reason about integrity properties.

Subtyping and Environment. Subtyping for keys is similar to the one for channels. Formally, the subtyping relation is defined as the least preorder such that:

$$\begin{array}{l} \ldots \text{ as in Equation 2} \\ LL \leq \mu\mathsf{K}^{LL}[LL, \ldots, LL] \\ \mu\mathsf{K}^{\ell}[\tilde{T}] \leq \ell \end{array} \tag{5}$$

$$\textsc{Empty}\quad \emptyset \vdash_{\mathcal{C}} \diamond \qquad \textsc{Env}\quad \frac{\Gamma \vdash_{\mathcal{C}} \diamond \qquad u \notin \mathit{dom}(\Gamma) \qquad T = \mu\mathsf{K}^{\ell}[\ldots], \mathsf{C}^{\ell}[\ldots] \Rightarrow \ell = HH, \mu \in \{\mathsf{Sym}, \mathsf{Dec}, \mathsf{Sig}\}}{\Gamma, u : T \vdash_{\mathcal{C}} \diamond}$$

Table 8. Cryptographic calculus: well-formedness of Γ

$$\textsc{EncKey}\quad \frac{\Gamma \vdash_{\mathcal{C}} K : \mathsf{DecK}^{\ell_C \ell_I}[\tilde{T}]}{\Gamma \vdash_{\mathcal{C}} \mathrm{ek}(K) : \mathsf{EncK}^{L\ell_I}[\tilde{T}]} \qquad \textsc{VerKey}\quad \frac{\Gamma \vdash_{\mathcal{C}} K : \mathsf{SigK}^{\ell_C \ell_I}[\tilde{T}]}{\Gamma \vdash_{\mathcal{C}} \mathrm{vk}(K) : \mathsf{VerK}^{L\ell_I}[\tilde{T}]}$$

$$\textsc{SymEnc}\quad \frac{\Gamma \vdash_{\mathcal{C}} K : \mathsf{SymK}^{\ell_C \ell_I}[\tilde{T}] \qquad \Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}}{\Gamma \vdash_{\mathcal{C}} \{|\tilde{M}|\}^{\mathsf{s}}_K : L\ell_I} \qquad \textsc{AsymEnc}\quad \frac{\Gamma \vdash_{\mathcal{C}} K : \mathsf{EncK}^{\ell_C \ell_I}[\tilde{T}] \qquad \Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}}{\Gamma \vdash_{\mathcal{C}} \{|\tilde{M}|\}^{\mathsf{a}}_K : L\ell_I}$$

$$\textsc{DigSig}\quad \frac{\Gamma \vdash_{\mathcal{C}} K : \mathsf{SigK}^{\ell_C \ell_I}[\tilde{T}] \qquad \Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T} \qquad \ell'_C = \sqcup_{T \in \tilde{T}} \mathcal{L}_{\mathrm{C}}(T)}{\Gamma \vdash_{\mathcal{C}} [\tilde{M}]_K : \ell'_C \ell_I}$$

Table 9. Cryptographic calculus: extending Table 4 to keys and cryptography.

Keys of level ℓ can be used in place of terms of type ℓ and terms of type LL can be used in place of keys of type $\mu\mathsf{K}^{LL}[LL, \ldots, LL]$. We denote this extended subtyping $\leq_{\mathcal{C}}$ to distinguish it from the core one. As expected, the level of a key type is ℓ, i.e., $\mathcal{L}(\mu\mathsf{K}^{\ell}[\tilde{T}]) = \ell$. As for the core type system, we have:

Remark 2 (Level subtyping for $\leq_{\mathcal{C}}$) $T \leq_{\mathcal{C}} T'$ implies $\mathcal{L}(T) \sqsubseteq \mathcal{L}(T')$.

The well-formedness of typing environments is defined in Table 8. Recall that we write u to denote a name or variable and we write $\vdash_{\mathcal{C}}$ to denote the new type system. Since encryption and verification keys are derived by their private counterparts, it is natural that their types are also derived. We thus allow in Γ only the types of symmetric, signing, and decryption keys. As for channels, only trusted HH keys are kept in Γ. Interestingly, as we have already shown in the example, derived keys will assume more articulate levels than just HH and LL, reflecting their asymmetric nature.

Typing Terms. In Table 9 we give the typing rules for the new terms, namely derived keys, encryptions and signatures. ENCKEY says that if a decryption key K is of type $\mathsf{DecK}^{\ell_C \ell_I}[\tilde{T}]$, then the corresponding encryption key $\mathrm{ek}(K)$ is of type $\mathsf{EncK}^{L\ell_I}[\tilde{T}]$. Notice that the confidentiality level is L, since public keys are allowed to be known to the attacker, while the integrity level is inherited from the decryption key; VERKEY does the same for verification and signing keys.

Ciphertexts are typed by SYMENC and ASYMENC. Ciphertexts can be output on public channels and consequently their confidentiality level is L. Their integrity level, instead, is the one of the key. The type of encrypted messages $\tilde{M}$ is required to be the one specified in the type of key. Digital signatures are typed by DIGSIG. The only difference with respect to the encryption rules is that the obtained confidentiality level is the maximum of the confidentiality levels of the signed messages $\tilde{M}$: these messages can be reconstructed from a signature using the public verification key, thus it is important to keep track of the confidentiality level of what is signed.

SYM DEC
$$\frac{\Gamma \vdash_{\mathcal{C}} M : T \qquad \Gamma \vdash_{\mathcal{C}} K : \mathsf{SymK}^{\ell}[\tilde{T}] \qquad \Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{C}} P}{\Gamma \vdash_{\mathcal{C}} \mathsf{case}\ M\ \mathsf{of}\ \{\!|\tilde{x}|\!\}^{\mathsf{s}}_{K}\ \mathsf{in}\ P}$$

ASYM DEC
$$\frac{\Gamma \vdash_{\mathcal{C}} M : T \qquad \Gamma \vdash_{\mathcal{C}} K : \mathsf{DecK}^{\ell}[\tilde{T}] \qquad \Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{C}} P \qquad \mathcal{L}_{\mathrm{I}}(T) = L \Rightarrow \Gamma, \tilde{x} : LL \vdash_{\mathcal{C}} P}{\Gamma \vdash_{\mathcal{C}} \mathsf{case}\ M\ \mathsf{of}\ \{\!|\tilde{x}|\!\}^{\mathsf{a}}_{K}\ \mathsf{in}\ P}$$

SIGN CHECK
$$\frac{\Gamma \vdash_{\mathcal{C}} M : T \qquad \Gamma \vdash_{\mathcal{C}} K : \mathsf{VerK}^{\ell_C \ell_I}[\tilde{T}] \qquad \Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{C}} P \qquad \mathcal{L}_{\mathrm{C}}(T) = H \Rightarrow \ell_I = H}{\Gamma \vdash_{\mathcal{C}} \mathsf{case}\ M\ \mathsf{of}\ [\tilde{x}]_{K}\ \mathsf{in}\ P}$$

NONCE CHECK
$$\frac{\Gamma \vdash_{\mathcal{C}} M : T \qquad n : T' \text{ in } \Gamma \qquad \mathcal{L}(T') \not\leq \mathcal{L}(T) \qquad \Gamma \vdash_{\mathcal{C}} P_2}{\Gamma \vdash_{\mathcal{C}} \mathsf{if}\ M = n\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2}$$

Table 10. Typing Rules for Processes

Characterizing Keys and Ciphertexts. We now review some important properties of keys and ciphertexts. As we will see, these properties are very close, in spirit, to the ones for channels (cf. section 2.2). As for high channels (Proposition 1), we show that the type of trusted HH keys is always in Γ, i.e., it can never be derived by a different type. In fact, only LL and LH keys can be respectively derived via SUBSUMPTION or ENCKEY/VERKEY.

Proposition 6 (High Keys for $\vdash_{\mathcal{C}}$) $\Gamma \vdash_{\mathcal{C}} N : \mu\mathsf{K}^{HH}[\tilde{T}]$ *implies* $N : \mu\mathsf{K}^{HH}[\tilde{T}]$ *in* Γ.

As for low channels (Proposition 2), keys of security level LL may only encrypt (or sign) messages of type LL.

Proposition 7 (Low Keys for $\vdash_{\mathcal{C}}$) $\Gamma \vdash_{\mathcal{C}} N : \mu\mathsf{K}^{LL}[\tilde{T}]$ *implies* $\tilde{T} = LL, \ldots, LL$.

Concerning the security level of keys, we have to make a distinction between private and public keys. Similarly to channels (Proposition 3), private keys can only be derived by ATOM or SUBSUMPTION, thus they can only assume levels LL and HH.

Proposition 8 (Private Keys for $\vdash_{\mathcal{C}}$) *If* $\Gamma \vdash_{\mathcal{C}} N : \mu\mathsf{K}^{\ell}[\tilde{T}]$ *and* $\mu \in \{\mathsf{Sym}, \mathsf{Sig}, \mathsf{Dec}\}$ *then* $\ell \in \{LL, HH\}$.

Public encryption/verification keys can only be derived via SUBSUMPTION, ENCKEY or VERKEY, but never from ATOM, from which the following result:

Proposition 9 (Public Keys for $\vdash_{\mathcal{C}}$) *If* $\Gamma \vdash_{\mathcal{C}} N : \mu\mathsf{K}^{\ell}[\tilde{T}]$ *and* $\mu \in \{\mathsf{Enc}, \mathsf{Ver}\}$ *then* $\ell \in \{LL, LH\}$.

Similarly to channels, the type of HH symmetric, decryption and signature keys is unique. For LL such keys, instead, we are not guaranteed of the uniqueness of μ as untrusted keys can be indifferently used as symmetric, decryption and signature keys. The transported type is instead guaranteed to be $LL, \ldots, LL$. We could also prove that the type of encryption and verification keys is unique if we fix the level to be LH. In fact, being them public, they can also be typed at level LL, reflecting their asymmetric nature. Since the latter property is not used in the proofs, we just state the former.

Proposition 10 (Uniqueness of Key Types for $\vdash_{\mathcal{C}}$) *If* $\Gamma \vdash_{\mathcal{C}} K : \mu\mathsf{K}^{\ell}[\tilde{T}]$ *and* $\Gamma \vdash_{\mathcal{C}} K : \mu'\mathsf{K}^{\ell'}[\tilde{T}']$ *with* $\mu, \mu' \in \{\mathsf{Sym}, \mathsf{Sig}, \mathsf{Dec}\}$ *and* $|\tilde{T}| = |\tilde{T}'|$ *then* $\ell = \ell'$ *and* $\tilde{T} = \tilde{T}'$. *If* $\ell = \ell' = HH$, *we also have* $\mu = \mu'$.

Finally, we characterize the type of encrypted (or signed) messages. Their type is dictated by the type of the private key, except for messages encrypted with asymmetric keys, which may also be of type LL if the ciphertext is low-integrity, e.g., received on an untrusted channel. In fact, the opponent can himself generate messages encrypted with honest principals public keys. For signatures we are also guaranteed that their confidentiality level is greater than or equal to the maximum confidentiality level of the signed messages. This is important to preserve the secrecy of signed terms.

Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$) *The following implications hold:*

1. $\Gamma \vdash_{\mathcal{C}} \{\!|\tilde{M}|\!\}^{\mathsf{s}}_{K} : T$ *and* $\Gamma \vdash_{\mathcal{C}} K : \mathsf{SymK}^{\ell}[\tilde{T}]$ *imply* $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}$.
2. $\Gamma \vdash_{\mathcal{C}} \{\!|\tilde{M}|\!\}^{\mathsf{a}}_{ek(K)} : T$ *and* $\Gamma \vdash_{\mathcal{C}} K : \mathsf{DecK}^{\ell}[\tilde{T}]$ *imply* $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}$ *or* $\mathcal{L}_{\mathrm{I}}(T) = L$ *and* $\Gamma \vdash_{\mathcal{C}} \tilde{M} : LL$.
3. $\Gamma \vdash_{\mathcal{C}} [\tilde{M}]_{K} : T$ *and* $\Gamma \vdash_{\mathcal{C}} K : \mathsf{SigK}^{\ell}[\tilde{T}]$ *imply* $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}$ *and* $\sqcup_{T_i \in \tilde{T}} \mathcal{L}_{\mathrm{C}}(T_i) \sqsubseteq_{\mathcal{C}} \mathcal{L}_{\mathrm{C}}(T)$.

Typing Processes. We finally extend the type system with the rules for processes performing cryptographic operations, as shown in Table 10. SYM DEC says that processes of the form $\mathsf{case}\ M\ \mathsf{of}\ \{\!|\tilde{x}|\!\}^{\mathsf{s}}_{K}\ \mathsf{in}\ P$, where K is a symmetric key of type $\mathsf{SymK}^{\ell}[\tilde{T}]$, are well-typed if M can be typed and P is well-typed in an environment where variables $\tilde{x}$ are given type $\tilde{T}$. This is sound since our type system guarantees that at run-time variables $\tilde{x}$ will only be replaced by values of type $\tilde{T}$. In fact, if the decryption succeeds, then M is a ciphertext of the form $\{\!|\tilde{M}|\!\}^{\mathsf{s}}_{K}$; since this term can only be typed by SYMENC and K has type $\mathsf{SymK}^{\ell}[\tilde{T}]$, we know that $\tilde{M}$ have types $\tilde{T}$.

ASYM DEC is close in spirit, but in the case of asymmetric cryptography we need to take into account that the encryption key is known to the attacker and therefore the ciphertext $\{\!|\tilde{M}|\!\}^{\mathsf{a}}_{\mathrm{ek}(K)}$ might come from the adversary meaning that $\tilde{M}$ could be of type LL. This might seem to be strange, since $\{\!|\tilde{M}|\!\}^{\mathsf{a}}_{K}$ can only be typed by ASYMENC and $\tilde{M}$ must have the type specified in the type of the encryption key $\mathrm{ek}(K)$. However, $\mathrm{ek}(K)$ can be given type $\mathsf{EncK}^{L\ell_I}[\tilde{T}]$ by ENCKEY as well as $\mathsf{EncK}^{LL}[LL, \ldots, LL]$ via the subtyping relation $\mathsf{EncK}^{L\ell_I}[\tilde{T}] \leq L\ell_I \leq LL \leq \mathsf{EncK}^{LL}[LL, \ldots, LL]$, which allows the attacker to type public encryption (and verification) keys. Since we cannot always statically predict if $\tilde{x}$ will be instantiated, at run-time, to values of type $\tilde{T}$ or values of type LL, we may have to type-check the continuation process twice, the first time under the assumption that the ciphertext comes from a honest participant, the second time under the assumption that the ciphertext comes from the attacker. In contrast to the type system proposed by Abadi and Blanchet in [3], where the continuation process is type-checked twice in any case, in our type system this happens only if the ciphertext is at low integrity, i.e., $\mathcal{L}_{\mathrm{I}}(T) = L$. As stated in Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$), if the ciphertext is at high integrity, we know that the type of encrypted messages is precisely the one specified in the key type and therefore we can simply type-check the continuation process under this typing assumption. This shows how combining integrity and confidentiality properties increases the precision of the analysis allowing us, e.g., to type-check processes based on the encrypt-then-sign paradigm, where the integrity of

the ciphertext is guaranteed by digital signature. An application of this rule is shown in Example 7.

SIGN CHECK is similar to SYM DEC but applies to processes performing the verification of a signature. The condition $\mathcal{L}_{\mathrm{C}}(T) = H \Rightarrow \ell_I = H$ is crucial for the soundness of our type system that combines confidentiality and integrity, since it avoids that processes use LL keys to verify a signature that transports high confidentiality data. In fact, that would downgrade the level to LL compromising secrecy.

Finally, NONCE CHECK is an additional rule, borrowed from [3] and adapted to fit our subtyping relation, that is typically used to type equality checks involving a nonce. Intuitively, since a name n can never be equal to a term M which is typed at a level which is not greater than or equal to the one of n, in such a case we can ignore the *then* branch of the equality test and we do not need to type-check it. This rule allows us to prune one of the typing branches introduced by ASYM DEC in the case the type of some of the messages in the ciphertext suffices to determine that the ciphertext does not come from the attacker. An application of this rule is illustrated in Example 8.

Example 6 We show that the Blanchet protocol of Example 4 and Example 5 is well typed, i.e., $A : LL, B : LL, c : LL \vdash_{\mathcal{C}}$ Protocol. We will prove that this guarantees secrecy and integrity of both the session key sent by B to A and the message sent by A to B. In the following, we focus on the typing rules applied for proving this judgment and illustrate how they modify the typing environment.

Rules Applied	Γ	$\vdash_{\mathcal{C}}$ Protocol
RES	$A : LL, B : LL, c : LL$	$(\nu k_A : \mathsf{DecK}^{HH}[HH])$
RES	$\ldots, k_A : \mathsf{DecK}^{HH}[HH]$	$(\nu k_B : \mathsf{SigK}^{HH}[LL, LL, T_k])$
PAR	$\ldots, k_B : \mathsf{SigK}^{HH}[LL, LL, T_k]$	(Initiator | Responder)

where T_k is $\mathsf{SymK}^{HH}[HH]$. The two restrictions just introduce the bindings $k_A : T_A$ and $k_B : T_B$. The same Γ is then used to independently type the initiator and the responder.

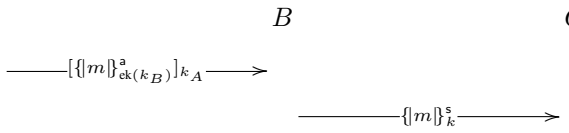
Rules Applied	Γ	$\vdash_{\mathcal{C}}$	Initiator
RES	$-$		$(\nu k : \mathsf{SymK}^{HH}[HH])$
OUT	$\ldots, k : \mathsf{SymK}^{HH}[HH]$		$\overline{c}\langle \{\!\lvert [A, B, k]_{k_B} \rvert\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)} \rangle$
IN	$-$		$c(x_e)$
ASYM DEC	$\ldots, x_e : LL$		$\mathsf{case}\ x_e\ \mathsf{of}\ \{\!\lvert x_m \rvert\!\}^{\mathsf{s}}_{k}\ \mathsf{in}\ \mathbf{0}$
STOP	$\ldots, x_m : HH$		$\mathbf{0}$

In the initiator process, the restriction of the session key is typed by RES, which introduces the type binding $k : \mathsf{SymK}^{HH}[HH]$. The output of $\{\!\lvert [A, B, k]_{k_B} \rvert\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)}$ is typed by OUT: in order to apply this rule, we have to prove that $[A, B, k]_{k_B}$ is of type HH (DIGSIG) and $\{\!\lvert [A, B, k]_{k_B} \rvert\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)}$ is of type LH (ASYMENC) and thus LL by subsumption. Notice that the ciphertext has to be given type LL since the type of c in Γ is indeed LL. The input of the response x_e is typed by IN, which introduces the type binding $x_e : LL$. The decryption of the ciphertext is typed by SYM DEC, which introduces the type binding $x_m : HH$, since HH is the type transported by key k. Process $\mathbf{0}$ is finally typed by STOP.

Rules Applied	Γ	$\vdash_C$	Responder
IN	$-$		$c(x_e).$
SYM DEC	$\ldots, x_e : LL$		case x_e of $\{\!\lvert x_s \rvert\!\}^{\mathsf{a}}_{k_A}$ in
SIGN CHECK	$\ldots, x_s : LL \,/\, \ldots, x_s : HH$		case x_s of $[x_A, x_B, x_k]_{\mathrm{vk}(k_B)}$ in
COND	$\ldots, x_A : LL, x_B : LL, x_k : T_k$		if $A = x_A$ then
RES	$-$		$(\nu m : HH)$
OUT	$\ldots, m : HH$		$\overline{c}\langle\{\!\lvert m \rvert\!\}^{\mathsf{s}}_{x_k}\rangle$

In the responder process, the input of the challenge is typed by IN, which introduces the type binding $x_e : LL$. The decryption of this message is typed by ASYM DEC: Since the ciphertext is of low confidentiality and we cannot statically determine if the ciphertext originates from the attacker or not, we have to type the continuation process twice, under the assumptions $x_s : LL$ and $x_s : HH$. The two typing derivations are, however, the same since x_s occurs only in the following signature check, which is typed by SIGN CHECK independently of the type of x_s. This rule introduces the bindings $x_A : LL, x_B : LL, x_k : T_k$, as specified in the type of k_B. The equality test is typed by COND and the generation of message m by RES. We finally type-check process $\overline{c}\langle\{\!\lvert m \rvert\!\}^{\mathsf{s}}_{x_k}\rangle$ by OUT, showing that $\{\!\lvert m \rvert\!\}^{\mathsf{s}}_{x_k}$ is of type LL by ASYMENC and subsumption. □

Example 7 As previously discussed, in our type system the process after an asymmetric decryption has to be type-checked twice only if the ciphertext is at low integrity. If the ciphertext is instead at high integrity, we type-check the continuation process only once, with the type information specified in the key, thus gaining precision in the analysis. We illustrate this technique on the following protocol:



A sends message m, encrypted and then signed, to B. B forwards this message to C, after encrypting it with a symmetric-key. The goal of this protocol is to guarantee the confidentiality and integrity of m. Notice that the two messages constitute different, but in a sense equivalent, cryptographic implementations of the (abstract) protocol shown in Example 3, which is based on ideal secure pi-calculus channels.

For typing this protocol, we give k_B type $\mathsf{DecK}^{HH}[HH]$, k_A type $\mathsf{SigK}^{HH}[LH]$, and the key shared between B and C type $\mathsf{SymK}^{HH}[HH]$. Notice that ciphertext $\{\!\lvert m \rvert\!\}^{\mathsf{a}}_{\mathrm{ek}(k_B)}$ is typed at level LH since it has been generated with the high integrity encryption key $\mathrm{ek}(K)$ of type $\mathsf{EncK}^{LH}[HH]$. It can thus be signed with key k_A. The obtained term is of type $LH \leq LL$ and can be sent on the network.

Intuitively, B knows that this ciphertext comes from A, since only A knows the signing key k_A and this key is only used in the protocol to sign encryptions of messages of level HH. Our type-system elegantly deals with this form of nested cryptography by giving the ciphertext $\{\!\lvert m \rvert\!\}^{\mathsf{a}}_{\mathrm{ek}(k_B)}$ obtained after signature verification type LH, as specified in the $\mathrm{vk}(k_A)$ type $\mathsf{VerK}^{LH}[LH]$. This allows us to give to decrypted message m type HH, as specified by the type of k_B. Notice that the high-integrity of the ciphertext, achieved via digital signature, allows us to type-check the continuation process once, with m bound to the correct type HH. B can thus encrypt m with symmetric key k of

type $\mathsf{SymK}^{HH}[HH]$. In the type system presented in [3], this protocol would not type-check since the process following the decryption would be type-checked a second time with $m : LL$. This would forbid the encryption with k since $LL \not\sqsubseteq HH$. □

Example 8 We now illustrate an interesting application of rule NONCE CHECK, which allows us to prune one of the typing branches introduced by ASYM DEC, in case the type of some of the decrypted messages suffices to determine that the ciphertext does not come from the attacker. Let us consider the Needham-Schroeder-Lowe protocol:

$$
\begin{array}{ll}
A & B \\
\longleftarrow\!\!\{\!|n_B, B|\!\}^{\mathsf{a}}_{\mathrm{ek}(k_A)}\!\!\longleftarrow & \\
\longrightarrow\!\!\{\!|n_A, n_B, A|\!\}^{\mathsf{a}}_{\mathrm{ek}(k_B)}\!\!\longrightarrow & \\
\longleftarrow\!\!\{\!|n_A|\!\}^{\mathsf{a}}_{\mathrm{ek}(k'_A)}\!\!\longleftarrow &
\end{array}
$$

For the sake of simplicity, our type system does not support types describing multiple usages of the same key. For type-checking this protocol, we have thus to assume that the first and the third ciphertext are encrypted using two different keys $\mathrm{ek}(k_A)$ and $\mathrm{ek}(k'_A)$.

B encrypts a fresh nonce n_B with A's public key and later receives a ciphertext encrypted with his own public key containing n_B, A's nonce n_A, and A's identifier. Let the two nonces n_A, n_B be given type HH and A's identifier type LL. Suppose the decryption of the second ciphertext binds these values to variables x_{n_A}, x_{n_B}, and x_A, respectively. Notice now that typing the decryption of the second message requires to type the continuation process twice, once with the expected payload types HH, HH, LL and once with types LL, LL, LL, to account for an encryption performed by the attacker. After decryption, however, B performs a nonce check by comparing x_{n_B} with n_B. By rule NONCE CHECK, we prune the case of x_{n_B} having type LL, since LL and the type HH of the nonce n_B are incomparable. Intuitively, the type system ensures that variables of type LL can only be instantiated to values of type $T \leq LL$ and therefore an equality check between a name of type HH and a variable of type LL will always fail at run-time, since $HH \not\leq LL$. □

Exercise 1 Model the Needham-Schroeder-Lowe protocol in the spi-calculus such that the resulting process type-checks. Give the two nonces type HH and provide suitable typing annotations for keys. (Hint: Pay attention to the type of n_B in the second encryption.)

Exercise 2 Extend the type system so to support multiple usages of the same key. This can be done by introducing the following type:

$$\mu\mathsf{K}^{\ell}[\tilde{T}_1 + \ldots + \tilde{T}_n]$$

This type describes keys used to encrypt tagged payloads $\mathrm{msg}_i(\tilde{M}_i)$ of type $\tilde{T}_i$, which is close in spirit to the tagged unions used in [23,24]. The calculus has to be extended with a term of the form $\{\!|\mathrm{msg}_i(\tilde{M})|\!\}^{\mathsf{a}}_K$ and a decryption primitive of the form $\mathsf{case}\ M\ \mathsf{of}\ \{\!|\mathrm{msg}_i(\tilde{x})|\!\}^{\mathsf{a}}_K\ \mathsf{in}\ P$, which checks at run-time that msg_i is the tag of the messages encrypted in M.

Properties of the Type System. We leave as an exercise to the interested reader the proof of strengthening, weakening, substitution, and opponent typability. We now prove subject congruence and subject reduction.

Proposition 12 (Subject Congruence and Reduction for $\vdash_{\mathcal{C}}$) *Let $\Gamma \vdash_{\mathcal{C}} P$. Then*

1. *$P \equiv Q$ implies $\Gamma \vdash_{\mathcal{C}} Q$;*
2. *$P \to Q$ implies $\Gamma \vdash_{\mathcal{C}} Q$.*

Proof:
1. The proof of subject congruence is the same as the one of Proposition 5 (Subject congruence and reduction)(1) as $\equiv$ is unchanged and the typing rules for processes related by $\equiv$ are also unchanged.

2. For proving subject reduction, we have to consider the typing rules introduced in Table 10 and the new reduction rules of Table 7.

For the NONCE CHECK rule, we know that $\Gamma \vdash_{\mathcal{C}} \mathsf{if}\ M = n\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$, $\Gamma \vdash_{\mathcal{C}} M : T$, $\Gamma(n) = T'$, $\mathcal{L}(T') \not\leq \mathcal{L}(T)$, and $\Gamma \vdash_{\mathcal{C}} P_2$. We prove that $\mathsf{if}\ M = n\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \not\to P_1$, i.e., by RED COND 1 $M \neq n$, which immediately proves the thesis since $\Gamma \vdash_{\mathcal{C}} P_2$. Let us assume by contradiction that $M = n$. Thus $\Gamma \vdash_{\mathcal{C}} n : T$, and $\Gamma(n) = T'$ which imply $T' \leq T$. By Remark 2 (Level subtyping for $\leq_{\mathcal{C}}$), $\mathcal{L}(T') \leq \mathcal{L}(T)$, which contradicts our hypothesis $\mathcal{L}(T') \not\leq \mathcal{L}(T)$.

We now consider the reduction rule of Table 7

$$\mathsf{case}\ \langle \tilde{M} \rangle_{K^+}\ \mathsf{of}\ \langle \tilde{x} \rangle_{K^-}\ \mathsf{in}\ P \quad \to \quad P\{\tilde{M}/\tilde{x}\}$$

By hypothesis, $\Gamma \vdash_{\mathcal{C}} \mathsf{case}\ \langle \tilde{M} \rangle_{K^+}\ \mathsf{of}\ \langle \tilde{x} \rangle_{K^-}\ \mathsf{in}\ P$. The three rules for proving this judgment are SYM DEC, ASYM DEC and SIGN CHECK. They all require $\Gamma \vdash_{\mathcal{C}} \langle \tilde{M} \rangle_{K^+} : T$ and $\Gamma \vdash_{\mathcal{C}} K^- : \mu\mathsf{K}^{\ell}[\tilde{T}]$ and $\Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{C}} P$, with $\mu = \mathsf{Sym}, \mathsf{Dec}, \mathsf{Ver}$, respectively. We examine the three different cases:

Symmetric decryption. By Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$)(1), we have $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}$. Since $\Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{C}} P$, by the substitution lemma, we obtain $\Gamma \vdash_{\mathcal{C}} P\{\tilde{M}/\tilde{x}\}$, as desired.

Asymmetric decryption. Rule ASYM DEC additionally requires variables to be typed LL, i.e., $\Gamma, \tilde{x} : LL \vdash_{\mathcal{C}} P$, when $\mathcal{L}_{\mathrm{I}}(T) = L$. By Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$)(2), we know that $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}$ or $\mathcal{L}_{\mathrm{I}}(T) = L \wedge \Gamma \vdash_{\mathcal{C}} \tilde{M} : LL$. Since $\Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{C}} P$ and $\Gamma, \tilde{x} : LL \vdash_{\mathcal{C}} P$ when $\mathcal{L}_{\mathrm{I}}(T) = L$, in both cases we can apply the substitution lemma and obtain $\Gamma \vdash_{\mathcal{C}} P\{\tilde{M}/\tilde{x}\}$, as desired.

Sign check. We cannot directly apply Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$)(3) since $\mu = \mathsf{Ver}$, i.e., $K^- = \mathrm{vk}(K)$ and $\Gamma \vdash_{\mathcal{C}} \mathrm{vk}(K) : \mathsf{VerK}^{\ell}[\tilde{T}]$. Proposition 9 (Public Keys for $\vdash_{\mathcal{C}}$) tells us that $\ell \in \{LL, LH\}$.

If $\ell = LH$, $\Gamma \vdash_{\mathcal{C}} \mathrm{vk}(K) : \mathsf{VerK}^{\ell}[\tilde{T}]$ can only derive from VERKEY, which implies $\Gamma \vdash_{\mathcal{C}} K : \mathsf{SigK}^{\ell_{\mathcal{C}} H}[\tilde{T}]$. By Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$)(3) we have $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T}$. By the substitution lemma, we obtain $\Gamma \vdash_{\mathcal{C}} P\{\tilde{M}/\tilde{x}\}$, as desired.

If, instead, $\ell = LL$, by Proposition 7 (Low Keys for $\vdash_{\mathcal{C}}$) we must have $\tilde{T} = LL, \ldots, LL$, and the judgment might derive either from VERKEY or from SUBSUMPTION. Anyway,

at some point of the derivation of $\Gamma \vdash_{\mathcal{C}} \mathrm{vk}(K) : \mathsf{VerK}^{LL}[\tilde{T}]$ we know it has been applied VERKEY as it is the only rule for typing term $\mathrm{vk}(K)$. This implies $\Gamma \vdash_{\mathcal{C}} K : \mathsf{SigK}^{\ell}[\tilde{T'}]$. Thus, by Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$)(3) we have $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T'}$ and $\mathcal{L}_{\mathrm{C}}(T) = \sqcup_{T' \in \tilde{T'}} \mathcal{L}_{\mathrm{C}}(T')$. Now recall that rule SIGN CHECK requires that $\mathcal{L}_{\mathrm{C}}(T) = H$ implies $\ell_I = H$. Since we have taken $\ell_I = L$ we know that $\mathcal{L}_{\mathrm{C}}(T) = L$. From $\mathcal{L}_{\mathrm{C}}(T) = \sqcup_{T' \in \tilde{T'}} \mathcal{L}_{\mathrm{C}}(T')$ we get $\forall T' \in \tilde{T'}, \mathcal{L}_{\mathrm{C}}(T') = L$ which implies $\mathcal{L}(T') \leq_{\mathcal{C}} LL$ and also $T' \leq_{\mathcal{C}} LL$ (since we always have $T \leq_{\mathcal{C}} \mathcal{L}(T)$). From $\Gamma \vdash_{\mathcal{C}} \tilde{M} : \tilde{T'}$ and SUBSUMPTION we thus get $\Gamma \vdash_{\mathcal{C}} \tilde{M} : LL$. By the substitution lemma, we obtain $\Gamma \vdash_{\mathcal{C}} P\{M/x\}$, as desired. □

Secrecy and Integrity of cryptographic protocols. The following lemma relates the previously defined semantic characterization of integrity to the notion of integrity captured in the type system: the integrity level of a typed message is always bounded by the integrity level of its type. In particular, messages with a high integrity type are shown to be at high integrity. In other words, the type system provides a sound overapproximation of the integrity level of messages.

Lemma 4 (Integrity) $\Gamma \vdash_{\mathcal{C}} M : T$ implies $\mathcal{L}_{\mathrm{I},\Gamma}(M) \sqsubseteq_I \mathcal{L}_{\mathrm{I}}(T)$.

The proof is left as an exercise to the reader. We can finally show that our type system statically enforces secrecy and integrity.

Theorem 2 (Secrecy and Integrity for $\vdash_{\mathcal{C}}$) *Let $\Gamma \vdash_{\mathcal{C}} P$ with $img(\Gamma) = \{LL\}$. Then P preserves both secrecy and integrity.*

Proof:
As for Theorem 1 (Secrecy and Integrity for $\vdash$) we pick an opponent O and we easily show that by extending Γ with the free names of O which are missing we obtain a Γ' such that $\Gamma' \vdash_{\mathcal{C}} P \mid O$. The proof of secrecy follows exactly the one of Theorem 1 (Secrecy and Integrity for $\vdash$). We thus focus on the integrity property. We first consider the following reduction:

$$P \mid Q \rightarrow^* (\nu k^+ : \mu\mathsf{K}^{HH}[T])\,(\nu \tilde{a} : \tilde{T})\,(P' \mid \mathsf{case}\ \langle M \rangle_{k^+}\ \mathsf{of}\ \langle x \rangle_{k^-}\ \ \mathsf{in}\ \ P'')$$

If $\mathcal{L}_{\mathrm{I}}(T) = L$ we have nothing to prove. Let thus $\mathcal{L}_{\mathrm{I}}(T) = H$. By Proposition 12 (Subject Congruence and Reduction for $\vdash_{\mathcal{C}}$)(2) we get

$$\Gamma' \vdash_{\mathcal{C}} (\nu k^+ : \mu\mathsf{K}^{HH}[T])\,(\nu \tilde{a} : \tilde{T})\,(P' \mid \mathsf{case}\ \langle M \rangle_{k^+}\ \mathsf{of}\ \langle x \rangle_{k^-}\ \ \mathsf{in}\ \ P'')$$

Let $\Gamma'' = \Gamma', k^+ : \mu\mathsf{K}^{HH}[T], \tilde{a} : \tilde{T}$. By repeatedly applying RES and finally by PAR we have: $\Gamma'' \vdash_{\mathcal{C}} \mathsf{case}\ \langle M \rangle_{k^+}\ \mathsf{of}\ \langle x \rangle_{k^-}\ \ \mathsf{in}\ \ P''$ and $\Gamma'' \vdash_{\mathcal{C}} \langle M \rangle_{k^+} : T$ for some T. Now notice that k^+ must be an atomic term since it is restricted, i.e., it cannot be $\mathrm{ek}(k)$ and thus $\langle M \rangle_{k^+} \neq \{\!|M|\!\}^{\mathsf{a}}_{k^+}$, and by $\Gamma'' \vdash_{\mathcal{C}} \diamond$ we have $\mu \in \{\mathsf{Sym}, \mathsf{Dec}, \mathsf{Sig}\}$. Thus, $\Gamma'' \vdash_{\mathcal{C}} \langle M \rangle_{k^+} : T$ implies $\Gamma'' \vdash_{\mathcal{C}} k^+ : \mu'\mathsf{K}^{\ell}[T']$, with $\mu' = \mathsf{Sym}$ or $\mu' = \mathsf{Sig}$ when $\langle M \rangle_{k^+}$ is a symmetric encryption or a digital signature, respectively.

Since $\Gamma'' \vdash_{\mathcal{C}} k^+ : \mu\mathsf{K}^{HH}[T]$, Proposition 10 (Uniqueness of Key Types for $\vdash_{\mathcal{C}}$) proves that $\mu = \mu'$, i.e., the μ in the restriction is coherent with the cryptographic operation. Therefore by Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$) we get $\Gamma'' \vdash_{\mathcal{C}} M : T$. Lemma 4 (Integrity) finally gives $\mathcal{L}_{\mathrm{I},\Gamma''}(M) \sqsubseteq_I \mathcal{L}_{\mathrm{I}}(T)$.

The proof for the other reduction:

$$P \mid O \rightarrow^* (\nu c : \mathsf{C}^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{c}\langle M \rangle.P'')$$

follows exactly the same steps as the one of Theorem 1 (Secrecy and Integrity for $\vdash$) to derive that $\Gamma'' \vdash_{\mathcal{C}} M : T$. By Lemma 4 (Integrity) we directly obtain $\mathcal{L}_{\mathrm{I},\Gamma''}(M) \sqsubseteq_I \mathcal{L}_{\mathrm{I}}(T)$. □

6. Authentication Protocols

In this section, we extend our type system in order to statically verify authentication protocols. Following the terminology introduced in [15], these protocols enable a party, called the *claimant*, to authenticate herself and possibly some messages with another party, called the *verifier*. In particular, we focus on a variant of the agreement property [28] that is well-suited to reason about authentication in cryptographic protocols based on nonce handshakes. The type system combines the main ideas of the type and effect systems for authentication protocols [23,24,30,15,8] with a more elegant formalism borrowed from the type systems for authorization policies [21,10].

6.1. Authentication

In this chapter, we consider the strongest of the authentication definitions proposed by Gavin Lowe in [28], namely *agreement*. Intuitively,

> "a protocol guarantees to a verifier A agreement with a claimant B on a set of data items ds if, whenever A (acting as verifier) completes a run of the protocol, apparently with claimant B, then B has previously been running the protocol, apparently with A, and B was acting as claimant in his run, and the two agents agreed on the data values corresponding to all the variables in ds, and each such run of A corresponds to a unique run of B."

In [28], the verifier and claimant are called initiator and responder, respectively. This property is formalized by annotating the point in the protocol where the claimant starts the authentication session (*begin assertion*) and the point in the protocol where the verifier accepts the authentication request (*end assertion*). These annotations are also known as correspondence assertions [38].

In this chapter, we focus on a variant of correspondence assertions introduced in [15], which is well-suited to reason about authentication protocols based on nonce handshakes. A nonce handshake is composed of two messages, the *challenge* sent by the verifier to the claimant and the *response* sent by the claimant to the verifier. Both the challenge and the response contain a random value (called *nonce*) freshly generated by the verifier: the nonce guarantees the freshness of the response, which entails the freshness of the authentication request. Of course, both the challenge and the response may contain, possibly encrypted or signed, other messages as well.

The syntax of processes is extended as follows:

$$\begin{array}{rll} P, Q, R, O ::= & \ldots & \text{(as in Table 6)} \\ & \mathrm{begin}_N(\tilde{M}; \tilde{N}) & \text{begin assertion} \\ & \mathrm{end}_N(\tilde{M}; \tilde{N}) & \text{end assertion} \end{array}$$

The begin and end assertions have three fundamental components: (i) the messages $\tilde{M}$ sent in the challenge, (ii) the messages $\tilde{N}$ sent in the response, and (iii) the nonce N. The agreement property is formalized as follows:

Definition 5 (Agreement) *A process P guarantees agreement if whenever $P \equiv (\nu\tilde{a} : \tilde{T})\ end_N(\widetilde{M}; \widetilde{N}) \mid Q$, we have that $Q \equiv begin_N(\widetilde{M}; \widetilde{N}) \mid Q'$ for some Q' and $(\nu\tilde{a} : \tilde{T})\ Q'$ guarantees agreement.*

Here and throughout this chapter we assume that replications are guarded and, in particular, they are never of the form $!P$ with $P \equiv (\nu\tilde{a} : \tilde{T})\ \mathrm{end}_N(\widetilde{M}; \widetilde{N}) \mid Q$. Otherwise the definition above would not be well-founded due to the unbounded number of top-level end assertions possibly introduced in the process via structural equivalence. We refine the notion of opponent by disallowing the presence of begin and end assertions. The former would break opponent typability, while the latter would vacuously break the agreement property even for safe protocols.

Definition 6 (Opponent) *A process O is an opponent if it contains neither begin nor end assertions and all $(\nu a : T)$ occurring therein are such that $T = LL$.*

We are interested in processes that guarantee agreement in the presence of arbitrary opponents. This property, called robust agreement, is stated below.

Definition 7 (Robust Agreement) *A process P guarantees robust agreement if for every opponent O and process Q such that $P \mid O \rightarrow^* Q$, Q guarantees agreement.*

Example 9 Let us consider again the Blanchet protocol of Example 4. This protocol is based on a nonce handshake between A and B, where B sends in the challenge a fresh session key k that is used by A to encrypt message m in the response. We decorate the code of the initiator as follows:

$$\begin{aligned}\text{Initiator} \triangleq\ & (\nu k : T_k)\ \overline{c}\langle \{\!|[A, B, k]_{k_B}|\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)}\rangle.c(x_e).\\ & \mathsf{case}\ x_e\ \mathsf{of}\ \{\!|x_m|\!\}^{\mathsf{s}}_{k}\ \ \mathsf{in}\ \ \mathrm{end}_k(A, B; x_m)\end{aligned}$$

For the moment let us ignore the typing annotation. The $\mathrm{end}_k(A, B; x_m)$ assertion says that B concludes an authentication session where he has sent the two identifiers A and B in the challenge and received message x_m in the response. The session key k guarantees the freshness of the authentication request, since each authentication session relies on a different key. In other words, in this protocol the session key plays the role of the nonce.

The process modelling the responder is reported below:

$$\begin{aligned}\text{Responder} \triangleq\ & c(x_e).\mathsf{case}\ x_e\ \mathsf{of}\ \{\!|x_s|\!\}^{\mathsf{a}}_{k_A}\ \ \mathsf{in}\ \ \mathsf{case}\ x_s\ \mathsf{of}\ [x_A, x_B, x_k]_{\mathrm{vk}(k_B)}\ \ \mathsf{in}\\ & \mathsf{if}\ A = x_A\ \mathsf{then}\ (\nu m : HH)\ \mathrm{begin}_{x_k}(x_A, x_B; m) \mid \overline{c}\langle \{\!|m|\!\}^{\mathsf{s}}_{x_k}\rangle\end{aligned}$$

The $\mathrm{begin}_{x_k}(x_A, x_B; m)$ assertion says that A confirms the reception of the identifiers x_A, x_B and declares the intention to authenticate the message m sent in the response. The session key x_k received in the challenge is supposed to guarantee the freshness of the authentication request. □

$$
\begin{array}{lll}
F, E & ::= & \mathrm{fresh}(N) \mid \mathrm{Chal}_N(\tilde{M}) \mid \mathrm{Resp}_N(\tilde{M}) \mid M = N \\
T & ::= & \ldots \text{ as in Equation 4} \\
 & \mid & \mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}] \qquad \text{(scope of } x, \tilde{x} \text{ is } \tilde{F}, \mathit{fnfv}(\tilde{F}) \subseteq \{x, \tilde{x}\}, \text{ and } \nexists N.\mathrm{fresh}(N) \in \tilde{F}) \\
\mu & ::= & x : T \mid F \\
\Gamma & ::= & \mu_1, \ldots, \mu_n
\end{array}
$$

Notation: $\mathit{dom}(\mu_1, \ldots, \mu_n) = \mathit{dom}(\mu_1) \cup \ldots \cup \mathit{dom}(\mu_n), \mathit{dom}(x : T) = \{x\}, \mathit{dom}(F) = \emptyset$.
$\mathit{eff}(\mu_1, \ldots, \mu_n) = \mathit{eff}(\mu_1) \cup \ldots \cup \mathit{eff}(\mu_n), \mathit{eff}(x : T) = \emptyset, \mathit{eff}(F) = \{F\}$
$\mathit{types}(\mu_1, \ldots, \mu_n) = \mathit{types}(\mu_1) \cup \ldots \cup \mathit{types}(\mu_n), \mathit{types}(x : T) = \{x : T\}, \mathit{types}(F) = \emptyset$

Table 11. Syntax of Types

6.2. Type System

Following [21,10], the typing environment is defined as a list of *effects* and type bindings. The syntax of effects and types is shown in Table 11.

The effects are similar to the ones used in [30,15,8]. The effect $\mathrm{fresh}(N)$ witnesses that N is a fresh nonce, i.e., it is restricted and it does not occur in any of the active end assertions. The effect $\mathrm{Chal}_N(\tilde{M})$ witnesses a challenge to authenticate messages $\tilde{M}$ with a fresh nonce N. In particular, if $\mathrm{Chal}_N(\tilde{M})$ belongs to the effect associated to a particular protocol point, then we know that at run-time, whenever that protocol point is reached, a party has sent (or is allowed to send) a challenge to authenticate $\tilde{M}$ with nonce N. Similarly, the effect $\mathrm{Resp}_N(\tilde{M})$ witnesses a response to authenticate messages $\tilde{M}$ with nonce N. Finally, $M = N$ witnesses that M is equal to N. Intuitively, the restriction of nonce N justifies $\mathrm{fresh}(N)$ and $\mathrm{Chal}_N(\tilde{M})$. The latter justifies a $\mathrm{begin}_N(\tilde{M}; \tilde{N})$ assertion, which in turn justifies $\mathrm{Resp}_N(\tilde{N})$. The effects $\mathrm{fresh}(N)$, $\mathrm{Chal}_N(\tilde{M})$, and $\mathrm{Resp}_N(\tilde{N})$ together justify the assertion $\mathrm{end}_N(\tilde{M}; \tilde{N})$, which annotates the end of a challenge-response protocol based on nonce N where $\tilde{M}$ have been sent in the challenge and $\tilde{N}$ in the response.

We introduce the new key type $\mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ that, besides the type of encrypted messages, describes their role in the challenge-response protocol. In particular, this type describes keys of security level ℓ that are used to encrypt a tuple $\tilde{x}$ of type $\tilde{T}$ such that the effects $\tilde{F}$ are justified. The scope of $x, \tilde{x}$ is $\tilde{F}$, where x is a binder for the (symmetric, encryption, or verification) key itself. We require that key types are closed (i.e., they do not contain free names or free variables) and do not contain fresh effects, since the freshness of a nonce is an information that is just used locally to type-check the process generating that nonce. In the following, we use $\mu\mathsf{K}^{\ell}[\tilde{T}]$ as an abbreviation for $\mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \emptyset]$. The subtyping relation is extended as expected:

$$
\begin{array}{l}
\ldots \text{ as in Equation 5} \\
\mu\mathsf{K}^{\ell}_{(\cdot)}[\ \ldots \mid \ldots\ ] \leq \ell \\
LL \leq \mu\mathsf{K}^{LL}_{(x)}[x_1 : LL, \ldots, x_n : LL \mid \emptyset]
\end{array}
$$

Keys of level ℓ can be used in place of values of type ℓ and, conversely, values of type LL can be used in place of LL keys that are supposed to encrypt LL messages for which no effect is justified, i.e., without providing any authentication guarantee.

$$\textsc{Empty}\quad \emptyset \vdash_{\mathcal{A}} \diamond$$

$$\textsc{A-Env}\quad \frac{\Gamma \vdash_{\mathcal{A}} \diamond \qquad u \notin \mathit{dom}(\Gamma) \qquad T = \mathsf{C}^{\ell}[\ldots], \mu\mathsf{K}^{\ell}_{(\cdot)}[\ldots \mid \ldots] \Rightarrow \ell = HH, \mu \in \{\mathsf{Sym}, \mathsf{Dec}, \mathsf{Sig}\}}{\Gamma, u : T \vdash_{\mathcal{A}} \diamond}$$

$$\textsc{A-Eff}\quad \frac{\Gamma \vdash_{\mathcal{A}} \diamond \qquad \mathit{fnfv}(F) \subseteq \mathit{dom}(\Gamma)}{\Gamma, F \vdash_{\mathcal{A}} \diamond}$$

Table 12. Well-formedness of Environments

The typing rules for the well-formedness of environments are shown in Table 12. A-ENV is the natural extension of ENV, while A-EFF says that an effect is well-formed if its free names and variables are bound in the typing environment.

Example 10 Let us consider again the Blanchet protocol described in Example 9. The types T_A and T_B of k_A and k_B are reported below:

$$\begin{aligned} T_A &\triangleq \mathsf{DecK}^{HH}[HH] \\ T_B &\triangleq \mathsf{SigK}^{HH}_{(z)}[z_A : LL, z_B : LL, z_k : T_k \mid \mathrm{Chal}_{z_k}(z_A, z_B)] \\ T_k &\triangleq \mathsf{SymK}^{HH}_{(z)}[z_m : HH \mid \mathrm{Resp}_z(z_m)] \end{aligned}$$

The two key-pairs are not revealed to and do not come from the attacker, hence their security level is HH. The type T_A says that the responder's key pair is used by well-typed parties only for encrypting messages at high confidentiality and high integrity, that is the signature generated by the initiator. The type T_B says that the initiator's key pair is only used to sign triples composed of two public and low integrity identifiers z_A and z_B and a session key z_k of type T_k and that the resulting signature constitutes a challenge from z_B to z_A in a handshake whose freshness is guaranteed by the freshness of the session key z_k. Finally, the type T_k says that the session key is confidential and high-integrity, it is only used to encrypt a secret and high-integrity message z_m, and the resulting ciphertext constitutes a response to authenticate z_m in a handshake whose freshness is guaranteed by the freshness of the session key itself. □

Typing Terms. The typing rules for terms are reported in Table 13. These rules are similar to the rules of Table 4. The main difference is that we check, before encrypting $\tilde{M}$ with a key K of type $\mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$, that the effects $\tilde{F}\{\tilde{M}/\tilde{x}, K'/x\}$ occur in the typing environment (cf. A-SYMENC, A-ASYMENC, A-SIGN). This is crucial for the soundness of the type system since the type of the key allows us to statically transfer effects from the sender to the receiver. As a matter of fact, the typing rules for decryption and signature verification extend the typing environment with the effects indicated in the key type. Notice that the key K' replacing the variable x is the key available to both the sender and the receiver, i.e., the symmetric key, the encryption key, or the verification key.

Characterizing Keys and Ciphertexts. We now see how the properties of keys and ciphertexts stated for type system $\vdash_{\mathcal{C}}$ can be smoothly extended to type system $\vdash_{\mathcal{A}}$. We first extend Proposition 6 (High Keys for $\vdash_{\mathcal{C}}$) to dependent key types.

$$\text{A-ATOM}\ \frac{\Gamma \vdash_{\mathcal{A}} \diamond \qquad M : T \text{ in } \Gamma}{\Gamma \vdash_{\mathcal{A}} M : T} \qquad \text{A-SUBSUMPTION}\ \frac{\Gamma \vdash_{\mathcal{A}} M : T' \qquad T' \leq T}{\Gamma \vdash_{\mathcal{A}} M : T} \qquad \text{A-ENCKEY}\ \frac{\Gamma \vdash_{\mathcal{A}} K : \mathsf{DecK}^{\ell_C \ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]}{\Gamma \vdash_{\mathcal{A}} \mathsf{ek}(K) : \mathsf{EncK}^{L\ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]}$$

$$\text{A-VERKEY}\ \frac{\Gamma \vdash_{\mathcal{A}} K : \mathsf{SigK}^{\ell_C \ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]}{\Gamma \vdash_{\mathcal{A}} \mathrm{vk}(K) : \mathsf{VerK}^{L\ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]}$$

$$\text{A-SYMENC}\ \frac{\Gamma \vdash_{\mathcal{A}} K : \mathsf{SymK}^{\ell_C \ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}] \qquad \Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T} \qquad \tilde{F}\{\tilde{M}/\tilde{x}, K/x\} \in \mathit{eff}(\Gamma)}{\Gamma \vdash_{\mathcal{A}} \{\!|\tilde{M}|\!\}^{\mathsf{s}}_K : L\ell_I}$$

$$\text{A-ASYMENC}\ \frac{\Gamma \vdash_{\mathcal{A}} K : \mathsf{EncK}^{\ell_C \ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}] \qquad \Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T} \qquad \tilde{F}\{\tilde{M}/\tilde{x}, K/x\} \in \mathit{eff}(\Gamma)}{\Gamma \vdash_{\mathcal{A}} \{\!|\tilde{M}|\!\}^{\mathsf{a}}_K : L\ell_I}$$

$$\text{A-SIGN}\ \frac{\begin{array}{c}\Gamma \vdash_{\mathcal{A}} K : \mathsf{SigK}^{\ell_C \ell_I}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}] \\ \Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T} \qquad \ell'_C = \sqcup_{T \in \tilde{T}} \mathcal{L}_{\mathrm{C}}(T) \qquad \tilde{F}\{\tilde{M}/\tilde{x}, \mathrm{vk}(K)/x\} \in \mathit{eff}(\Gamma)\end{array}}{\Gamma \vdash_{\mathcal{A}} [\tilde{M}]_K : \ell'_C \ell_I}$$

Notation: $F_1, \ldots, F_m \in \mathit{eff}(\Gamma)$ iff $\forall i \in [1, n], F_i \in \mathit{eff}(\Gamma)$.

Table 13. Typing Rules for Terms

Proposition 13 (High Keys for $\vdash_{\mathcal{A}}$) $\Gamma \vdash_{\mathcal{A}} M : \mu\mathsf{K}^{HH}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ *implies* $M : \mu\mathsf{K}^{HH}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ *is in* Γ.

We then extend Proposition 7 (Low Keys for $\vdash_{\mathcal{C}}$) by stating that low-level keys do not provide any authentication guarantees.

Proposition 14 (Low Keys for $\vdash_{\mathcal{A}}$) $\Gamma \vdash_{\mathcal{A}} N : \mu\mathsf{K}^{LL}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ *implies* $\tilde{T} = LL, \ldots, LL$ *and* $\tilde{F} = \emptyset$.

The next two lemmas are the direct counterpart of Proposition 8 (Private Keys for $\vdash_{\mathcal{C}}$) and Proposition 9 (Public Keys for $\vdash_{\mathcal{C}}$).

Proposition 15 (Private Keys for $\vdash_{\mathcal{A}}$) $\Gamma \vdash_{\mathcal{A}} N : \mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ *with* $\mu \in \{\mathsf{Sym}, \mathsf{Sig}, \mathsf{Dec}\}$ *implies* $\ell \in \{LL, HH\}$.

Proposition 16 (Public Keys for $\vdash_{\mathcal{A}}$) $\Gamma \vdash_{\mathcal{A}} N : \mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ *with* $\mu \in \{\mathsf{Enc}, \mathsf{Ver}\}$ *implies* $\ell \in \{LL, LH\}$.

An important property of our type system is that channels as well as private keys have a unique type. Additionally, the typing of channels does not depend on effects.

Proposition 17 (Uniqueness of Channel types for $\vdash_{\mathcal{A}}$) *If* $\Gamma \vdash_{\mathcal{A}} N : \mathsf{C}^{\ell}[\tilde{T}]$ *and* $\Gamma' \vdash_{\mathcal{A}} N : \mathsf{C}^{\ell'}[\tilde{T'}]$ *with types*$(\Gamma) =$ *types*(Γ') *and* $|\tilde{T}| = |\tilde{T'}|$ *then* $\mathsf{C}^{\ell}[\tilde{T}] = \mathsf{C}^{\ell'}[\tilde{T'}]$.

Proposition 18 (Uniqueness of Key Types for $\vdash_{\mathcal{A}}$) *If* $\Gamma \vdash_{\mathcal{C}} K : \mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ *and* $\Gamma \vdash_{\mathcal{C}} K : \mu'\mathsf{K}^{\ell'}_{(x)}[\tilde{x} : \tilde{T'} \mid \tilde{F'}]$ *with* $\mu \in \{\mathsf{Sym}, \mathsf{Sig}, \mathsf{Dec}\}$ *and* $|\tilde{T}| = |\tilde{T'}|$ *then* $\ell = \ell'$, $\tilde{T} = \tilde{T'}$, *and* $\tilde{F} = \tilde{F'}$. *When* $\ell = \ell' = HH$, *we also have* $\mu = \mu'$.

Finally, we characterize the type of encrypted (or signed) messages, in the same style as Proposition 11 (Payload Type for $\vdash_{\mathcal{C}}$). Notice that the effects in the key type must belong to the typing environment used to type-check the ciphertext (or the signature).

Proposition 19 (Payload Type for $\vdash_{\mathcal{A}}$) *The following implications hold:*

1. *If* $\Gamma \vdash_{\mathcal{A}} \{|\tilde{M}|\}^{\mathsf{s}}_{K} : T$ *and* $\Gamma \vdash_{\mathcal{A}} K : \mathsf{SymK}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$*, then* $\Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$ *and* $\tilde{F}\{K/x, \tilde{M}/\tilde{x}\} \in$ *eff*(Γ).
2. *If* $\Gamma \vdash_{\mathcal{A}} \{|\tilde{M}|\}^{\mathsf{a}}_{ek(K)} : T$ *and* $\Gamma \vdash_{\mathcal{A}} K : \mathsf{DecK}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$*, then* $\Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$ *and* $\tilde{F}\{ek(K)/x, \tilde{M}/\tilde{x}\} \in$ *eff*(Γ)*, or* $\mathcal{L}_{\mathrm{I}}(T) = L$ *and* $\Gamma \vdash_{\mathcal{A}} \tilde{M} : LL$.
3. *If* $\Gamma \vdash_{\mathcal{A}} [\tilde{M}]_{K} : T$ *and* $\Gamma \vdash_{\mathcal{A}} K : \mathsf{SigK}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$*, then* $\Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$ *and* $\tilde{F} \in \{vk(K)/x, \tilde{M}/\tilde{x}\} \in$ *eff*(Γ) *and* $\mathcal{L}_{\mathrm{C}}(T) = \sqcup_{T \in \tilde{T}} \mathcal{L}_{\mathrm{C}}(T)$.

Typing Processes The typing rules for processes are reported in Table 14. The main difference with respect to the rules of Table 10 is the part related to the effects, since the type bindings are managed in the same way.

A-STOP checks the well-formedness of the typing environment. A-PAR says that the parallel composition $P \mid Q$ is well typed in the typing environment Γ if P and Q are well-typed in typing environments obtained from Γ by partitioning the freshness effects. This is crucial to ensure that each nonce is used in at most one end assertion. In addition, the typing environment used to type-check P also contains the response effects justified by the top-level begin assertions in Q (and vice-versa). This technical detail is important, since the begin assertions do not have a continuation process and the response effect they justify has to be propagated to the processes in parallel composition. Function $\overline{P}$ returns the response atomic effects justified by the top-level begin assertions in P but the effects containing names restricted in P. Function $\Gamma_{|E}$ returns the projection of Γ to type bindings and effects in E.

A-REPL checks that the typing environment contains no fresh effects, since we want to prevent multiple end assertions with the same nonce. A-RES says that the restriction $(\nu a : T)\, P$ justifies the effects $\mathrm{Chal}_a(\tilde{M}), \mathrm{fresh}(a)$ in the continuation process P. Notice that the type system is not syntax-directed since A-RES can add an arbitrary prefix of $\mathrm{Chal}_a(\tilde{M}), \mathrm{fresh}(a)$ to the typing environment and the $\tilde{M}$'s are non-deterministically chosen.

A-IN, and A-OUT do not modify the effects. A-SYMDEC, A-ASYMDEC, A-SIGCHECK say that the decryption (or signature verification) $\mathsf{case}\ M\ \mathsf{of}\ \langle \tilde{x} \rangle_{K^-}\ \mathsf{in}\ P$ with a key of type $\mu\mathsf{K}^{\ell}_{(y)}[\tilde{y} : \tilde{T} \mid \tilde{F}]$ justifies the effect $\tilde{F}\{K'/y, \tilde{x}/\tilde{y}\}$ in the continuation process P, where, as discussed before, K' is the key available to both the sender and the receiver, i.e., the symmetric key, the encryption key, and the verification key, respectively.

A-COND type-checks the equality test $\mathsf{if}\ M = N\ \mathsf{then}\ P$, justifying in the typing environment of the continuation process the equality $M = N$. A-BEGIN type-checks the begin assertion $\mathrm{begin}_N(\tilde{M}; \tilde{N})$, requiring that the terms $\tilde{M}$ have indeed been received in a challenge with nonce N (i.e., $\mathrm{Chal}_N(\tilde{M})$ belongs to the typing environment). Similarly, A-END type-checks the end assertion $\mathrm{end}_n(\tilde{M}; \tilde{N})$, requiring that the terms $\tilde{M}$ have been sent in a challenge with nonce n, the terms $\tilde{N}$ have been received in a response with nonce n, and n is fresh (i.e., the typing environment contains the effects $\mathrm{Chal}_n(\tilde{M})$, $\mathrm{Resp}_N(\tilde{N})$, and $\mathrm{fresh}(n)$ as well as effects proving the equality between n and N).

A-STOP
$$\frac{\Gamma \vdash_{\mathcal{A}} \diamond}{\Gamma \vdash_{\mathcal{A}} \mathbf{0}}$$

A-PAR
$$\frac{\Gamma_{|E_P}, \overline{Q} \vdash_{\mathcal{A}} P \qquad \Gamma_{|E_Q}, \overline{P} \vdash_{\mathcal{A}} Q \qquad E_P \cup E_Q = \mathit{eff}(\Gamma) \qquad E_P \cap E_Q = \{f \in \mathit{eff}(\Gamma) \mid \nexists N.f = \mathrm{fresh}(N)\}}{\Gamma \vdash_{\mathcal{A}} P \mid Q}$$

A-REPL
$$\frac{\Gamma \vdash_{\mathcal{A}} P \qquad \nexists N.\mathrm{fresh}(N) \in \mathit{eff}(\Gamma)}{\Gamma \vdash_{\mathcal{A}} !P}$$

A-RES
$$\frac{\Gamma, a : T, \Gamma' \vdash_{\mathcal{A}} P \qquad \Gamma' \preceq \mathrm{Chal}_a(\tilde{M}), \mathrm{fresh}(a)}{\Gamma \vdash_{\mathcal{A}} (\nu a : T)\ P}$$

A-IN
$$\frac{\Gamma, \tilde{x} : \tilde{T} \vdash_{\mathcal{A}} P \qquad \Gamma \vdash_{\mathcal{A}} N : \mathsf{C}^{\ell}[\tilde{T}]}{\Gamma \vdash_{\mathcal{A}} N(\tilde{x}).P}$$

A-OUT
$$\frac{\Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T} \qquad \Gamma \vdash_{\mathcal{A}} P \qquad \Gamma \vdash_{\mathcal{A}} N : \mathsf{C}^{\ell}[\tilde{T}]}{\Gamma \vdash_{\mathcal{A}} \overline{N}\langle \tilde{M} \rangle.P}$$

A-SYMDEC
$$\frac{\Gamma \vdash_{\mathcal{A}} M : T \qquad \Gamma \vdash_{\mathcal{A}} K : \mathsf{SymK}^{\ell}_{(y)}[\tilde{y} : \tilde{T} \mid \tilde{F}] \qquad \Gamma, \tilde{x} : \tilde{T}, \tilde{F}\{K/y, \tilde{x}/\tilde{y}\} \vdash_{\mathcal{A}} P}{\Gamma \vdash_{\mathcal{A}} \mathsf{case}\ M\ \mathsf{of}\ \{\!|\tilde{x}|\!\}^{\mathsf{s}}_{K}\ \mathsf{in}\ P}$$

A-ASYMDEC
$$\frac{\Gamma \vdash_{\mathcal{A}} M : T \qquad \Gamma \vdash_{\mathcal{A}} K : \mathsf{DecK}^{\ell}_{(y)}[\tilde{y} : \tilde{T} \mid \tilde{F}] \qquad \Gamma, \tilde{x} : \tilde{T}, \tilde{F}\{\mathsf{ek}(K)/y, \tilde{x}/\tilde{y}\} \vdash_{\mathcal{A}} P \qquad \mathcal{L}_{\mathrm{I}}(T) = L \Rightarrow \Gamma, \tilde{x} : LL \vdash_{\mathcal{A}} P}{\Gamma \vdash_{\mathcal{A}} \mathsf{case}\ M\ \mathsf{of}\ \{\!|\tilde{x}|\!\}^{\mathsf{a}}_{K}\ \mathsf{in}\ P}$$

A-SIGCHECK
$$\frac{\Gamma \vdash_{\mathcal{A}} M : T \qquad \Gamma \vdash_{\mathcal{A}} K : \mathsf{VerK}^{\ell}_{(y)}[\tilde{y} : \tilde{T} \mid \tilde{F}] \qquad \Gamma, \tilde{x} : \tilde{T}, \tilde{F}\{K/y, \tilde{x}/\tilde{y}\} \vdash_{\mathcal{A}} P \qquad \mathcal{L}_{\mathrm{C}}(T) = H \Rightarrow \ell_I = H}{\Gamma \vdash_{\mathcal{A}} \mathsf{case}\ M\ \mathsf{of}\ [\tilde{x}]_K\ \mathsf{in}\ P}$$

A-COND
$$\frac{\Gamma \vdash_{\mathcal{A}} M : T \qquad \Gamma \vdash_{\mathcal{A}} N : T' \qquad \Gamma, M = N \vdash_{\mathcal{A}} P \qquad \Gamma \vdash_{\mathcal{A}} Q}{\Gamma \vdash_{\mathcal{A}} \mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q}$$

A-NONCE CHECK
$$\frac{\Gamma \vdash_{\mathcal{A}} M : T \qquad \Gamma(n) = T' \qquad \mathcal{L}(T') \not\leq \mathcal{L}(T) \qquad \Gamma \vdash_{\mathcal{A}} P_2}{\Gamma \vdash_{\mathcal{A}} \mathsf{if}\ M = n\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2}$$

A-BEGIN
$$\frac{\mathrm{Chal}_N(\tilde{M}) \in \mathit{eff}(\Gamma) \qquad \mathit{fnfv}(\tilde{N}) \subseteq \mathit{dom}(\Gamma) \qquad \Gamma \vdash_{\mathcal{A}} \diamond}{\Gamma \vdash_{\mathcal{A}} \mathrm{begin}_N(\tilde{M}; \tilde{N})}$$

A-END
$$\frac{n =_{\Gamma} N \qquad \mathrm{fresh}(n), \mathrm{Chal}_n(\tilde{M}), \mathrm{Resp}_N(\tilde{N}) \in \mathit{eff}(\Gamma) \qquad \Gamma \vdash_{\mathcal{A}} \diamond}{\Gamma \vdash_{\mathcal{A}} \mathrm{end}_n(\tilde{M}; \tilde{N})}$$

Notation:

$\overline{P} = \{\mathrm{Resp}_N(\tilde{N}) \text{ s.t. } P \equiv \mathrm{begin}_N(\tilde{M}; \tilde{N}) \mid P'\}$.

$\emptyset_{|E} = \emptyset; (\Gamma, x : T)_{|E} = \Gamma_{|E}, x : T; (\Gamma, F)_{|E} = \Gamma_{|E}$ if $F \notin E$; $(\Gamma, F)_{|E} = \Gamma_{|E}, F$ if $F \in E$.

$\Gamma \preceq \Gamma'$ iff Γ is a, possibly empty, prefix of Γ'.

$=_{\Gamma}$ is the smallest equivalence relation on terms such that if $N = M \in \Gamma$ then $N =_{\Gamma} M$.

Table 14. Typing Rules for Processes.

Example 11 We prove that the Blanchet protocol is well typed, i.e., $A : LL, B : LL, c : LL \vdash_{\mathcal{A}}$ Protocol. In the following, we illustrate the typing derivation, focusing just on the effects and omitting the type bindings, since the latter are exactly as in Example 6.

Rules Applied	*eff*(Γ)	$\vdash_{\mathcal{A}}$ Protocol
A-RES	$\emptyset$	$(\nu k_A : T_A)$
A-RES	$\emptyset$	$(\nu k_B : T_B)$
A-PAR	$\emptyset$	(Initiator | Responder)

The two restrictions do not involve nonces, so they do not increase the effect.

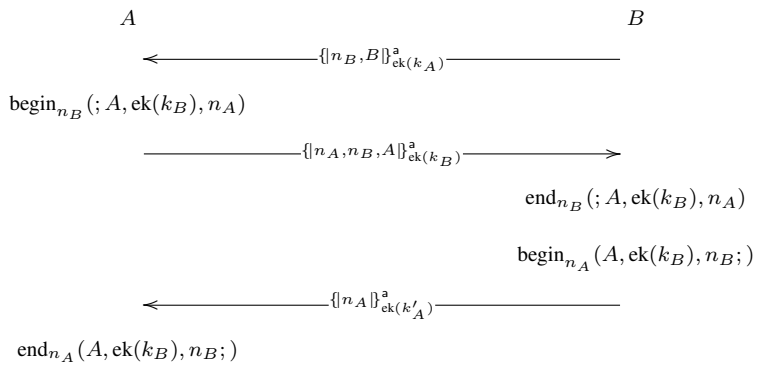
Rules Applied	*eff*(Γ)	$\vdash_{\mathcal{A}}$ Initiator
A-RES	–	$(\nu k : T_k)$
A-OUT	$\mathrm{fresh}(k), \mathrm{Chal}_k(A, B)$	$\overline{c}\langle \{\!\lvert [A, B, k]_{k_B} \rvert\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)} \rangle$
A-IN	–	$c(x_e)$
A-SYMDEC	–	case x_e of $\{\!\lvert x_m \rvert\!\}^{\mathsf{s}}_{k}$ in
A-END	$\ldots, \mathrm{Resp}_k(x_m)$	$\mathrm{end}_k(A, B; x_m)$

In the initiator process, the restriction of the session key is typed by A-RES, which introduces the effects $\mathrm{fresh}(k)$ and $\mathrm{Chal}_k(A, B)$. The output of $\{\!\lvert [A, B, k]_{k_B} \rvert\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)}$ is typed by A-OUT: in order to apply this rule, we have to prove that $[A, B, k]_{k_B}$ is of type HH (A-SIGN) and $\{\!\lvert [A, B, k]_{k_B} \rvert\!\}^{\mathsf{a}}_{\mathsf{ek}(k_A)}$ is of type LH (A-ASYMENC) and thus LL by subsumption. This requires that $\mathrm{Chal}_k(A, B)$ belongs to the current typing environment, since the signing key has type $\mathsf{SigK}^{HH}_{(z)}[z_A : LL, z_B : LL, z_k : T_k \mid \mathrm{Chal}_{z_k}(z_A, z_B)]$. The input of the response x_e is typed by A-IN, which does not change the effects. The decryption of the ciphertext is typed by A-SYMDEC, which introduces the response effect $\mathrm{Resp}_k(x_m)$ obtained from the type $\mathsf{SymK}^{HH}_{(z)}[z_m : HH \mid \mathrm{Resp}_z(z_m)]$ of the session key after replacement of the variables z and z_m by k and x_m, respectively. Since the typing environment contains the effects $\mathrm{fresh}(k)$, $\mathrm{Chal}_k(A, B)$, and $\mathrm{Resp}_k(x_m)$, the assertion $\mathrm{end}_k(A, B; x_m)$ can be typed by A-END.

Rules Applied	*eff*(Γ)	$\vdash_{\mathcal{A}}$ Responder
A-IN	–	$c(x_e)$.
A-ASYMDEC	–	case x_e of $\{\!\lvert x_s \rvert\!\}^{\mathsf{a}}_{k_A}$ in
A-SIGCHECK	–	case x_s of $[x_A, x_B, x_k]_{\mathrm{vk}(k_B)}$ in
A-COND	$\mathrm{Chal}_{x_k}(x_A, x_B)$	if $A = x_A$ then
A-RES	$\ldots, A = x_A$	$(\nu m : HH)$
A-BEGIN	–	$\mathrm{begin}_{x_k}(x_A, x_B; m)$ |
A-OUT	$\ldots, \mathrm{Resp}_{x_k}(m)$	$\overline{c}\langle \{\!\lvert m \rvert\!\}^{\mathsf{s}}_{x_k} \rangle$

In the responder process, the input of the challenge and the decryption do not modify the effect. A-SIGCHECK introduces the effect $\mathrm{Chal}_{x_k}(x_A, x_B)$, as specified in the type $\mathsf{VerK}^{LH}_{(z)}[z_A : LL, z_B : LL, z_k : T_k \mid \mathrm{Chal}_{z_k}(z_A, z_B)]$ of $\mathrm{vk}(k_B)$. The equality test is typed by A-COND and the generation of the message to authenticate by A-RES. Since the typing environment contains the effect $\mathrm{Chal}_{x_k}(x_A, x_B)$, we can type-check $\mathrm{begin}_{x_k}(x_A, x_B; m)$ by A-BEGIN. Rule A-Par allows us to exploit the effect $\mathrm{Resp}_{x_k}(m)$ derived from the begin assertion when typing $\overline{c}\langle \{\!\lvert m \rvert\!\}^{\mathsf{s}}_{x_k} \rangle$. A-OUT is applied by showing that $\{\!\lvert m \rvert\!\}^{\mathsf{s}}_{x_k}$ is of type LL (by A-ASYMENC and subsumption), which in turn requires that the effect $\mathrm{Resp}_{x_k}(m)$ belongs to the current typing environment. □

Exercise 3 Model and type-check the Needham-Schroeder-Lowe protocol annotated as follows:



These assertions model mutual authentication between A and B on the two nonces n_A and n_B. Notice that the public encryption key $\mathrm{ek}(k_B)$ takes the role of B's identifier in the correspondence assertions, since B's identifier is not sent in the second message and in our type system the dependent key types must be closed (i.e., the effects occurring therein can solely depend on encrypted terms and encryption keys). In fact, in the second ciphertext B is identified via his own public key.

6.3. Properties of the Type System

We extend the strengthening lemma to effects by showing that removing duplicate effects as well as effects containing names or variables not occurring in $\mathcal{J}$ preserves the typability of $\mathcal{J}$.

Lemma 5 (Strengthening for $\vdash_{\mathcal{A}}$) The following properties hold:

1. If $\Gamma, M : T, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ and $M \notin \mathit{fnfv}(\mathcal{J}) \cup \mathit{fnfv}(\Gamma')$, then $\Gamma; \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$.
2. If $\Gamma, F, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ and $F \in \mathit{eff}(\Gamma, \Gamma')$, then $\Gamma, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$.
3. If $\Gamma, F, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ and $\mathit{fnfv}(F) \not\subseteq \mathit{fnfv}(\mathcal{J})$, then $\Gamma, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$.

The weakening lemma allows us to arbitrarily extend the typing environment as long as we do not introduce fresh atomic effects. Extensions with fresh atomic effects would, for instance, prevent us from type-checking replications.

Lemma 6 (Weakening for $\vdash_{\mathcal{A}}$) The following properties hold:

1. If $\Gamma, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ and $\Gamma, x : T, \Gamma' \vdash_{\mathcal{A}} \diamond$ then $\Gamma, x : T, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$.
2. If $\Gamma, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ and $\Gamma, F, \Gamma' \vdash_{\mathcal{A}} \diamond$ and $\nexists N.F = \mathrm{fresh}(N)$ then $\Gamma, F, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$.

The substitution lemma is stated below. Notice that the substitution applies also to the typing environment because of dependent types and effects.

Lemma 7 (Substitution for $\vdash_{\mathcal{A}}$) If $\Gamma \vdash_{\mathcal{A}} M : T$, then $\Gamma, x : T, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ implies $\Gamma, \Gamma'\{M/x\} \vdash_{\mathcal{A}} \mathcal{J}\{M/x\}$.

The proofs of these properties and the proof of opponent typability are left as an exercise to the interested reader. An important property of our type system is that removing fresh effects from the typing environment does not affect the typing of terms. This property is used in the proof of subject reduction.

Lemma 8 (Fresh and Terms) If $\Gamma, \mathrm{fresh}(N), \Gamma' \vdash_{\mathcal{A}} M : T$ then $\Gamma, \Gamma' \vdash_{\mathcal{A}} M : T$.

Proof:
By induction on the derivation of $\Gamma, \mathrm{fresh}(N), \Gamma' \vdash_{\mathcal{A}} M : T$ and in the proof of the A-SYMENC, A-ASYMENC and A-SIGN cases by observing that, by syntactic restriction, $\Gamma \vdash K : \mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ implies $\nexists N.\mathrm{fresh}(N) \in \tilde{F}$. □

Finally, we show that type bindings and effects can be swapped as long as the well-formedness of the typing environment is preserved.

Lemma 9 (Exchange) If $\Gamma, \mu, \mu', \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$ and $\mathit{dom}(\mu) \cap \mathit{fnfv}(\mu') = \emptyset$, then $\Gamma, \mu', \mu, \Gamma' \vdash_{\mathcal{A}} \mathcal{J}$.

With this setup in place, we can finally prove subject congruence and subject reduction. We have, however, to make an important change in the semantics of the calculus, i.e., we remove rule $(\nu a : T)\ (\nu b : T')\ P \equiv (\nu b : T')\ (\nu a : T)\ P$ from the definition of the structural equivalence relation. When the type system is dependent, this rule is problematic since T' might depend on a, thus breaking subject congruence. A common solution to this problem is indeed to forbid the exchange of restrictions (see, e.g., [24,10]). An alternative solution is possible when the type annotations capture all the dependencies introduced in the typing environment: in this case, we can keep the rule and additionally require $a \notin \mathit{fn}(T')$ (cf. [23]). This is not possible in our type system since the challenge effects introduced by A-RES are not captured by the typing annotations.

Proposition 20 (Subject Congruence and Reduction for $\vdash_{\mathcal{A}}$) *Let $\Gamma \vdash_{\mathcal{A}} P$. Then*

1. *$P \equiv Q$ implies $\Gamma \vdash_{\mathcal{A}} Q$;*
2. *$P \to Q$ implies $\Gamma \vdash_{\mathcal{A}} Q$.*

Proof:
1. The only interesting case is when $(\nu a : T)\ (P \mid Q) \equiv P \mid (\nu a : T)\ Q$ with $a \notin \mathit{fn}(P)$. We have $\Gamma, a : T, \Gamma' \vdash_{\mathcal{A}} P \mid Q$, with $\Gamma' \preceq \mathrm{Chal}_a(\tilde{M}), \mathrm{fresh}(a)$, which can only be proved by A-PAR. We thus have $(\Gamma, a : T, \Gamma')_{|E_P}, \overline{Q} \vdash_{\mathcal{A}} P$ and $(\Gamma, a : T, \Gamma')_{|E_Q}, \overline{P} \vdash_{\mathcal{A}} Q$, with $E_P \cup E_Q = \mathit{eff}(\Gamma, \Gamma')$ and $E_P \cap E_Q = \{f \in \mathit{eff}(\Gamma, \Gamma') \mid \nexists N.f = \mathrm{fresh}(N)\}$.

Since $a \notin \mathit{fn}(P)$, by applying Lemma 5 (Strengthening for $\vdash_{\mathcal{A}}$) (3), we get $\Gamma_{|E_P}, \overline{(\nu a : T)\ Q} \vdash_{\mathcal{A}} P$. Notice that $\overline{(\nu a : T)\ Q}$ removes from $\overline{Q}$ all the response effects with occurrences of a.

Since $a \notin \mathit{fn}(P)$ and thus $a \notin \mathit{fn}(\overline{P})$, we can apply Lemma 9 (Exchange) to get $\Gamma_{|E_Q}, \overline{P}, a : T, \Gamma'_{|E_Q} \vdash_{\mathcal{A}} Q$. By A-RES, we obtain $\Gamma_{|E_Q}, \overline{P} \vdash_{\mathcal{A}} (\nu a : T)\ Q$. By A-PAR, we finally get $\Gamma \vdash_{\mathcal{A}} P \mid (\nu a : T)\ Q$.

Conversely, $\Gamma \vdash_{\mathcal{A}} P \mid (\nu a : T)\ Q$ can only be proved by A-PAR, which implies $\Gamma_{|E_P}, \overline{(\nu a : T)\ Q} \vdash_{\mathcal{A}} P$ and $\Gamma_{|E_Q}, \overline{P} \vdash_{\mathcal{A}} (\nu a : T)\ Q$, with $E_P \cup E_Q = \mathit{eff}(\Gamma)$ and $E_P \cap E_Q = \{f \in \mathit{eff}(\Gamma) \mid \nexists N.f = \mathrm{fresh}(N)\}$.

The judgment $\Gamma_{|E_Q}, \overline{P} \vdash_{\mathcal{A}} (\nu a : T)\, Q$ can only be proved by A-RES, which implies $\Gamma_{|E_Q}, \overline{P}, a : T, \Gamma' \vdash_{\mathcal{A}} Q$, with $\Gamma' \preceq \mathrm{Chal}_a(\tilde{M}), \mathrm{fresh}(a)$.

Since $a \notin \mathit{fn}(P)$, we have $a \notin \mathit{fn}(\overline{P})$. Therefore, by Lemma 9 (Exchange), we also have $\Gamma_{|E_Q}, a : T, \Gamma', \overline{P} \vdash_{\mathcal{A}} Q$

Since $a \notin \mathit{fn}(P)$, we can apply Lemma 6 (Weakening for $\vdash_{\mathcal{A}}$) to add $a : T$ and $\mathrm{Chal}_a(\tilde{M})$ and $\overline{Q} \setminus \overline{(\nu a : T)\, Q}$ to the typing environment used to type-check P, thus obtaining $\Gamma_{|E_P}, a : T, \Gamma'_{|\{\mathrm{Chal}_a(\tilde{M})\}}, \overline{Q} \vdash_{\mathcal{A}} P$ from $\Gamma_{|E_P}, \overline{(\nu a : T)\, Q} \vdash_{\mathcal{A}} P$.

Notice that $\mathrm{fresh}(a)$ is possibly used for typing Q, but it is not used when typing P. We can therefore apply A-PAR to get $\Gamma, a : T, \Gamma' \vdash_{\mathcal{A}} P \mid Q$. By A-RES, we finally obtain $\Gamma \vdash_{\mathcal{A}} (\nu a : T)\, P \mid Q$.

2. The proof is by induction on the derivation of $P \to Q$ and by case analysis of the last applied rule. We prove the interesting cases below:

(RED RES) Straightforward, by A-RES and induction hypothesis.

(RED PAR) We know that $P \mid R \to Q \mid R$ is derived from $P \to Q$ and we have $\Gamma \vdash_{\mathcal{A}} P \mid R$. This typing judgment can only be proved by A-PAR, which implies $\Gamma_{|E_P}, \overline{R} \vdash_{\mathcal{A}} P$ and $\Gamma_{|E_R}, \overline{P} \vdash_{\mathcal{A}} R$ with $E_P \cup E_R = \mathit{eff}(\Gamma)$ and $E_P \cap E_R = \{f \in \mathit{eff}(\Gamma) \mid \nexists N. f = \mathrm{fresh}(N)\}$.

By induction hypothesis, $\Gamma_{|E_P}, \overline{R} \vdash_{\mathcal{A}} Q$. By an inspection of rule A-BEGIN and by observing that $\Gamma \vdash_{\mathcal{A}} \mathcal{J}$ implies $\Gamma \vdash \diamond$, we can easily see that $\overline{P} \subseteq \overline{Q}$ and $\mathit{fn}(\overline{Q}) \subseteq \mathit{dom}(\Gamma)$. By Lemma 6 (Weakening for $\vdash_{\mathcal{A}}$), we get $\Gamma_{|E_R}, \overline{Q} \vdash_{\mathcal{A}} R$. By A-PAR, we finally obtain $\Gamma \vdash_{\mathcal{A}} Q \mid R$.

RED I/O We have $\overline{N}\langle\tilde{M}\rangle.P \mid N(\tilde{x}).Q \to P \mid Q\{\tilde{M}/\tilde{x}\}$ and $\Gamma \vdash_{\mathcal{A}} \overline{N}\langle\tilde{M}\rangle.P \mid N(\tilde{x}).Q$.

Since this typing judgment can only by proved by A-PAR and $\overline{\overline{N}\langle\tilde{M}\rangle.P} = \overline{N(\tilde{x}).Q} = \emptyset$, we have $\Gamma_{|E_P} \vdash_{\mathcal{A}} \overline{N}\langle\tilde{M}\rangle.P$ and $\Gamma_{|E_Q} \vdash_{\mathcal{A}} N(\tilde{x}).Q$.

By A-Out, we must have $\Gamma_{|E_P} \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$ with $\Gamma_{|E_P} \vdash_{\mathcal{A}} N : \mathsf{C}^{\ell}[\tilde{T}]$ and $\Gamma_{|E_P} \vdash_{\mathcal{A}} P$.

By A-IN and Proposition 17 (Uniqueness of Channel types for $\vdash_{\mathcal{A}}$), we must have $\Gamma_{|E_Q} \vdash N : \mathsf{C}^{\ell}[\tilde{T}]$ and $\Gamma_{|E_Q}, \tilde{x} : \tilde{T} \vdash_{\mathcal{A}} Q$.

Since $E_P \cap E_Q = \{f \in \mathit{eff}(\Gamma) \mid \nexists N. f = \mathrm{fresh}(N)\}$ and $\Gamma_{|E_P} \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$, by Lemma 8 (Fresh and Terms) and Lemma 6 (Weakening for $\vdash_{\mathcal{A}}$) we obtain $\Gamma_{|E_Q} \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$. By Lemma 7 (Substitution for $\vdash_{\mathcal{A}}$), $\Gamma_{|E_Q} \vdash_{\mathcal{A}} Q\{\tilde{M}/\tilde{x}\}$.

As before, we can see that $\mathit{fn}(\overline{P}) \subseteq \mathit{dom}(\Gamma)$ and $\mathit{fnfv}(\overline{Q}) \subseteq \mathit{dom}(\Gamma)$. By Lemma 6 (Weakening for $\vdash_{\mathcal{A}}$), we get $\Gamma_{|E_P}, \overline{Q} \vdash_{\mathcal{A}} P$ and $\Gamma_{|E_Q}, \overline{P} \vdash_{\mathcal{A}} Q\{\tilde{M}/\tilde{x}\}$. By PAR, we get $\Gamma \vdash_{\mathcal{A}} P \mid Q\{\tilde{M}/\tilde{x}\}$.

(RED DEC/CHECK) We have $\mathsf{case}\ \langle\tilde{M}\rangle_{K^+}\ \mathsf{of}\ \langle\tilde{y}\rangle_{K^-}\ \mathsf{in}\ P \quad \to \quad P\{\tilde{M}/\tilde{y}\}$.

By hypothesis, $\Gamma \vdash_{\mathcal{A}} \mathsf{case}\ \langle\tilde{M}\rangle_{K^+}\ \mathsf{of}\ \langle\tilde{y}\rangle_{K^-}\ \mathsf{in}\ P$. The three rules for proving this judgment are A-SYMDEC, A-ASYMDEC and A-SIGCHECK. They all require $\Gamma \vdash_{\mathcal{A}} \langle\tilde{M}\rangle_{K^+} : T$ and $\Gamma \vdash_{\mathcal{A}} K^- : \mu\mathsf{K}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid \tilde{F}]$ and $\Gamma, \tilde{y} : \tilde{T}, \tilde{F}\{K'/x, \tilde{y}/\tilde{x}\} \vdash_{\mathcal{A}} P$, with $\mu = \mathsf{Sym}, \mathsf{Dec}, \mathsf{Ver}$ and $K' = K, K^+, K^-$, respectively.

We examine in detail the case of symmetric decryption, which follows similarly to the corresponding case in the proof of Proposition 12 (Subject Congruence and Reduction for $\vdash_{\mathcal{C}}$). The proof for the other cases is similar.

Symmetric decryption By Proposition 19 (Payload Type for $\vdash_{\mathcal{A}}$)(1), we have $\Gamma \vdash_{\mathcal{A}} \tilde{M} : \tilde{T}$ and $\tilde{F}\{K/x, \tilde{M}/\tilde{y}\} \in \mathit{eff}(E)$.
Since $\Gamma, \tilde{y} : \tilde{T}, \tilde{F}\{K'/x, \tilde{y}/\tilde{x}\} \vdash_{\mathcal{A}} P$, by Lemma 7 (Substitution for $\vdash_{\mathcal{A}}$) we obtain $\Gamma, \tilde{F}\{K/x, \tilde{M}/\tilde{x}\} \vdash_{\mathcal{A}} P\{\tilde{M}/\tilde{y}\}$.
Since $\tilde{F}\{K/x, \tilde{M}/\tilde{y}\} \in \mathit{eff}(E)$, by Lemma 5 (Strengthening for $\vdash_{\mathcal{A}}$) (2) we obtain $\Gamma \vdash_{\mathcal{A}} P\{\tilde{M}/\tilde{y}\}$.

□

We can finally state the main result of our analysis technique, i.e., well-typed processes guarantee robust agreement.

Theorem 3 (Robust Agreement) *Let $\Gamma \vdash_{\mathcal{A}} P$ with $\mathit{img}(\Gamma) = \{LL\}$. Then P guarantees robust agreement.*

Proof:
We have to prove that for every Q and opponent O such that $P \mid O \rightarrow^* Q$, Q guarantees agreement.

As for Theorem 1 (Secrecy and Integrity for $\vdash$), we can show that by extending Γ with the free names of O that are missing we obtain a Γ' such that $\Gamma' \vdash_{\mathcal{C}} P \mid O$.

By Proposition 20 (Subject Congruence and Reduction for $\vdash_{\mathcal{A}}$), we have $\Gamma' \vdash_{\mathcal{A}} Q$. If $Q \equiv (\nu \tilde{a} : \tilde{T})\ \mathrm{end}_N(\tilde{M}; \tilde{N}) \mid Q'$, then $\mathrm{Chal}_N(\tilde{M}), \mathrm{Resp}_N(\tilde{N})$ and $\mathrm{fresh}(N)$ belong to the typing environment used to type-check the end assertion (cf. rule A-END). The presence of the fresh effect implies $N \in \widetilde{a}$. The presence of the response effect implies $Q' \equiv \mathrm{begin}_N(\tilde{M'}; \tilde{N}) \mid Q''$ for some $\tilde{M'}, Q''$. We must have that $\mathrm{Chal}_N(\tilde{M'})$ belongs to the typing environment used to type-check the begin assertion. Since the $\widetilde{a}$'s are pairwise distinct (otherwise the typing environment used to type-check the end assertion would not be well-formed), we have that the two challenge effects derive from the same restriction and are thus the same, implying $\tilde{M} = \tilde{M'}$.

We also know that there are no other occurrences of $\mathrm{end}_N(\tilde{M}; \tilde{N})$ in Q'', since each of them would require a typing environment containing $\mathrm{fresh}(N)$, but the typing rule for parallel composition requires that the typing environments of the two parallel processes do not share any single fresh effect. This implies that Q guarantees agreement.

□

Exercise 4 Extend the type system with dependent channel types of the form $\mathsf{C}^{\ell}_{(x)}[\tilde{x} : \tilde{T} \mid F]$ and show that such an extension preserves the properties studied in this section.

References

[1] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.

[2] M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. In *Proc. 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2030 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2001.

[3] M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. In *Proc. 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2030 of *Lecture Notes in Computer Science*, pages 25–41. Springer-Verlag, 2001.

[4] M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. *Theoretical Computer Science*, 298(3):387–415, 2003.

[5] M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1):102–146, 2005.

[6] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[7] A. Askarov, D. Hedin, and A. Sabelfeld. Cryptographically-masked flows. *Theoretical Computer Science*, 402(2-3):82–101, August 2008.

[8] M. Backes, A. Cortesi, R. Focardi, and M. Maffei. A calculus of challenges and responses. In *Proc. 5th ACM Workshop on Formal Methods in Security Engineering (FMSE)*, pages 101–116. ACM Press, 2007.

[9] M. Backes, C. Hriţcu, and M. Maffei. Type-checking zero-knowledge. In *15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 357–370. ACM Press, 2008.

[10] J. Bengtson, K. Bhargavan, C. Fournet, A. D. Gordon, and S. Maffeis. Refinement types for secure implementations. In *Proc. 21th IEEE Symposium on Computer Security Foundations (CSF)*, pages 17–32. IEEE Computer Society Press, 2008.

[11] B. Blanchet. Automatic verification of security protocols: formal model and computational model, 2008. Habilitation thesis.

[12] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. R. Nielson. Static validation of security protocols. *Journal of Computer Security*, 13(3):347–390, 2005.

[13] M. Bugliesi, R. Focardi, and M. Maffei. Authenticity by tagging and typing. In *Proc. 2nd ACM Workshop on Formal Methods in Security Engineering (FMSE)*, pages 1–12. ACM Press, 2004.

[14] M. Bugliesi, R. Focardi, and M. Maffei. Analysis of typed-based analyses of authentication protocols. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW)*, pages 112–125. IEEE Computer Society Press, 2005.

[15] M. Bugliesi, R. Focardi, and M. Maffei. Dynamic types for authentication. *Journal of Computer Security*, 15(6):563–617, 2007.

[16] M. Centenaro and R. Focardi. Information flow security of multi-threaded distributed programs. In *ACM SIGPLAN PLAS'08*, pages 113–124, June 8 2008.

[17] M. Centenaro, R. Focardi, F. Luccio, and G. Steel. Type-based Analysis of PIN Processing APIs. In *14th European Symposium on Research in Computer Security (ESORICS'09)*, LNCS, September 2009. To appear.

[18] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th Symposium on Principles of Programming Languages (POPL)*, pages 238–252. ACM Press, 1977.

[19] R. Focardi and M. Maffei. Types for security protocols. Technical Report CS-2010-3, University of Venice, 2010. Available at `http://www.lbs.cs.uni-saarland.de/resources/types-security.pdf`.

[20] C. Fournet, A. D. Gordon, and S. Maffeis. A type discipline for authorization policies. In *Proc. 14th European Symposium on Programming (ESOP)*, Lecture Notes in Computer Science, pages 141–156. Springer-Verlag, 2005.

[21] C. Fournet, A. D. Gordon, and S. Maffeis. A type discipline for authorization in distributed systems. In *Proc. 20th IEEE Symposium on Computer Security Foundations (CSF)*, pages 31–45. IEEE Computer Society Press, 2007.

[22] C. Fournet and T. Rezk. Cryptographically sound implementations for typed information-flow security. In *POPL'08*, pages 323–335. ACM Press, 2008.

[23] A. D. Gordon and A. Jeffrey. Authenticity by typing for security protocols. *Journal of Computer Security*, 11(4):451–519, 2003.

[24] A. D. Gordon and A. Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3):435–484, 2004.

[25] A. D. Gordon and A. Jeffrey. Secrecy despite compromise: Types, cryptography, and the pi-calculus. In *Proc. 16th International Conference on Concurrency Theory (CONCUR)*, volume 3653, pages 186–201. Springer-Verlag, 2005.

[26] C. Haack and A. Jeffrey. Timed spi-calculus with types for secrecy and authenticity. In *Proc. 16th International Conference on Concurrency Theory (CONCUR)*, volume 3653, pages 202–216. Springer-Verlag, 2005.

[27] P. Laud. On the computational soundness of cryptographically masked flows. In *POPL'08*, pages 337–348. ACM Press, January 10-12 2008.

[28] G. Lowe. "A Hierarchy of Authentication Specifications". In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE Computer Society Press, 1997.

[29] M. Maffei. Tags for multi-protocol authentication. In *Proc. 2nd International Workshop on Security Issues in Coordination Models, Languages, and Systems (SECCO '04)*, Electronic Notes on Theoretical Computer Science, pages 55–63. Elsevier Science Publishers Ltd., 2004.

[30] M. Maffei. *Dynamic Typing for Security Protocols*. PhD thesis, Universita Ca' Foscari di Venezia, Dipartimento di Informatica, 2006.

[31] R. Milner. The polyadic pi-calculus: a tutorial. In *Logic and Algebra of Specification*, pages 203–246, 1993.

[32] J. Morris. Protection in programming languages. *Communications of the ACM*, 16(1):15–21, 1973.

[33] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.

[34] A. Sabelfeld and A. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, January 2003.

[35] E. Sumii and B. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1-3):169–192, 2007.

[36] E. Sumii and B. C. Pierce. Logical relations for encryption. *Journal of Computer Security*, 11(4):521–554, 2003.

[37] J. Vaughan and S. Zdancewic. A cryptographic decentralized label model. In *IEEE Symposium on Security and Privacy*, pages 192–206. IEEE Computer Society, 2007.

[38] T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operation Systems Review*, 28(3):24–37, 1994.