A Calculus of Challenges and Responses

Michael Backes[1], Agostino Cortesi[2], Riccardo Focardi[2], and Matteo Maffei[1]

[1] Saarland University, Saarbrücken, Germany, {backes,maffei}@cs.uni-sb.de
[2] Ca' Foscari University, Venice, Italy, {cortesi,focardi}@dsi.unive.it

Abstract. This paper presents a novel approach for concisely abstracting authentication protocols and for subsequently analyzing those abstractions in a sound manner, i.e., deriving authentication guarantees for protocol abstractions suffices for proving these guarantees for the actual protocols. The abstractions are formalized in a novel calculus which constitutes a higher-level abstraction of the ρ-spi calculus and is specifically tailored towards reasoning about challenge-response mechanisms within authentication protocols. Furthermore, it allows for expressing protocols without having to include details on the specific structure of exchanged messages. This in particular entails that many authentication protocols share a common abstraction so that a single validation of this abstraction already gives rise to security guarantees for all these protocols. Such an abstract validation can be automatically performed using static analysis techniques based on an effect system proposed in this paper. Finally, extensions to abstractions of additional protocol classes immediately enjoy a soundness theorem provided that these extensions satisfy certain explicit, easily checkable conditions.

1 Introduction

Language-based security has proved to be a salient technique for formally analyzing security protocols, since Abadi's seminal work on secrecy by typing [1] up to modern techniques based on logics, model-checking and type systems (a far from being comprehensive list includes [3, 20, 16, 13, 18, 10, 14]). Authentication protocols are known to require very subtle reasoning about encrypting and decrypting various related messages in the presence of a powerful adversary, with each of these encryptions being meant to contribute some part of the overall authentication guarantee. Thus they are strongly vulnerable to attacks originating by a certain degree of ambiguity in the protocol message, e.g., man-in-the-middle attacks or attacks where certain encryptions are re-used by the adversary in another protocol execution where they suddenly get a different semantics. This turns authentication protocols into particularly suitable targets of language-based security, and several current language-based static analysis techniques [18, 10, 14] for tackling this problem have been proposed, e.g., based on protocols narrations formalized in the spi-calculus [4], or in variants thereof such as Lysa [9] and the ρ-spi calculus [13]. Roughly speaking, these techniques typically rely on some static patterns, defined on the syntax of the calculus, which suffice for showing that the run-time protocol execution respects certain intended challenge-response schemes which in turn imply the desired authentication guarantees.

For instance, the type system by Gordon and Jeffrey [18] relies on some type information provided by the user, which suffices for identifying nonces and formalizing to which extent their exchange contributes to achieving authentication. The generality of the analysis is sometimes paid by sophisticated type definitions that have to be defined manually and thus require a certain degree of expertise from the programmers. The type system by Bugliesi, Focardi and Maffei [14] relies on some dynamic information attached to ciphertexts, in the form of tags, which univocally determine the role of messages in the authentication task. A great advantage is that tag and type definitions are automatically inferred [22]. Furthermore, the use of tags makes the analysis compositional, thus naturally fitting the analysis of multi-protocol systems, where participants engage in different, and possibly unknown, protocols. Even if the analysis is general enough to cover several existing protocols, its scope is strictly constrained by the set of tagged ciphertexts.

This paper tackles the problem of analyzing authentication protocols from a novel and conceptually more abstract perspective. Starting from protocol narrations expressed in a dialect of the ρ-spi calculus, we abstract from the specific structure of messages by solely focussing on the challenge-response components that are inherent in the protocols. The resulting more abstract protocols are formalized in a new process calculus, the CR *calculus*, which is specifically tailored to reasoning about challenge-response mechanisms within authentication protocols. The CR calculus grants a conceptually more abstract view of authentication protocols, enables more abstract and general proofs, and enjoys some properties constituting a significant advantage over existing approaches.

– Foremost, we prove a soundness theorem stating that abstractions preserve authentication properties, i.e., proving authentication for a protocol abstraction in the CR calculus suffices for obtaining an authentication proof for the actual ρ-spi protocol.
– The abstraction induces a uniform representation of different authentication protocols as it does not have to reflect details on the specific structure of messages; in fact, many authentication protocols share a common abstraction so that a security proof for this abstraction guarantees the security of all these protocols.
– The analysis is modular and compositional since each principal is independently validated and the parallel composition of successfully validated protocols yields a secure protocol again. This fits very well the analysis of multi-protocol systems.
– The abstraction from ρ-spi to CR calculus protocol descriptions and the effect system used for verifying the safety of the latter are completely independent. This independence constitutes a key feature of our approach since it entails that refining the abstraction does not affect the soundness of the analysis and viceversa.
– The abstraction is extensible in that extensions to additional protocol classes immediately enjoy a soundness theorem provided that these extensions satisfy certain explicit, easily checkable conditions. This is a key difference with respect to other type-based approaches [2, 18, 14], where an extension of the analysis requires reproving the soundness and the safety of the analysis from scratch.

Further related work. The control-flow analysis for message authenticity proposed by Bodei *et al.* in [10] and the abstract interpretation for distributed systems and, as specific instance, for cryptographic protocols discussed by Feret in [17] are closely related to our approach: however, they address message authenticity and not the freshness of authentication requests.

Table 1 (Our Dialect of) the ρ-spi Calculus

Names		**Terms**		**Processes**	
$a ::= n, m$	(Msg)	$C ::= a$	(Name)	$P, Q ::= \mathsf{new}(n).P$	(Res)
I, J, A, B, E	(Id)	x, y, z	(Var)	$\mathsf{in}(C).P$	(In)
Keys		$\mathsf{Tag}(C)$	(Tag)	$\mathsf{out}(C).P$	(Out)
$k ::= k_{IJ}$	(Sym)	(C_1, C_2)	(Pair)	$\mathsf{begin}_N(A, I, M_1; M_2).P$	(Begin)
k_I^+, k_I^-	(Asym)	$\{\mid C \mid\}_k$	(Enc)	$\mathsf{end}_N(A, J, M_1; M_2).P$	(End)
				$A \triangleright P$	(Princ)
Notation: M, N denote terms without encryptions and tags.				$P \mid Q$	(Par)
				$!P$	(Repl)
				$\mathbf{0}$	(Stop)

The strand spaces formalism [19, 20] constitutes an interesting framework for studying privacy and authenticity: even if authors provide some effective patterns for deriving the proofs of safety, these are in general written by hand. ProVerif is a tool implementing an automatic analysis based on logic programming [7, 3, 8], which has been proved very effective in the analysis of security protocols. In contrast to dynamic typing, these techniques prove safety results against participants running the protocol of interest. However, as discussed in [23], problems may arise when participants execute different protocols with the same cryptographic keys. The interaction among *different* protocols, possibly unknown or, by contrast, carrying out some common sub-tasks, is particularly interesting in a global computing setting, where several security services coexist and are possibly combined together. Interestingly, strand spaces offer some syntactic conditions on protocol specifications for guaranteeing the compositionality of the analysis: this still assumes that protocols interacting with each other are known.

Finally, we would like to mention a logic-based approach to cryptographic protocol analysis proposed by Durgin *et al.* [16]. Relying on protocol invariants, they develop a modular way of reasoning about security protocols, ensuring that protocols that are proved to be individually secure do not interact insecurely when they are composed with other protocols. Since the analysis technique and the underlying model is different, a formal comparison is interesting but laborious and is left as future work.

Due to space constraints, some technical aspects are only outlined in this paper: we refer the interested reader to the long version [6].

Outline. Section 2 reviews the ρ-spi calculus and introduces a small novel dialect thereof. Section 3 presents the new CR calculus. Section 4 introduces the abstraction of ρ-spi protocol descriptions into the CR calculus. Section 5 proposes an effect system for checking the safety of CR protocols. Section 6 concludes and outlines future work.

2 Review of the ρ-spi Calculus, and a Novel Dialect Thereof

The ρ-spi calculus [13] is derived from the spi calculus [4] and inherits many of the features of *Lysa* [9], a dialect of the spi calculus specifically tailored to the analysis of authentication protocols. The ρ-spi calculus differs from both calculi in several respects: it incorporates the notion of tagged message exchange from [12], associates principal identities to processes and syntactically binds keys to their owners. In this paper, we consider a novel dialect of ρ-spi in which encryptions and decryptions are performed on-the-fly when sending and receiving messages, respectively. This dialect in particular links protocol specifications more tightly to their informal "graphical" descriptions, which only depict sent and received messages without giving a precise semantics on how messages are parsed and constructed.

2.1 Syntax of our Dialect of the ρ-spi Calculus

The formal syntax of our dialect of the ρ-spi calculus is depicted in Table 1. We presuppose a countable set $\mathcal{N}$ of *names* partitioned into two distinct categories: *messages* and *identities*. The set of identities $\mathcal{ID}$, ranged over by I and J, is further partitioned into *trusted principals* $\mathcal{ID}_{\mathcal{P}}$, ranged over by A and B, and *enemies* $\mathcal{ID}_{\mathcal{E}}$, ranged over by E. *Keys* are partitioned into symmetric keys k_{IJ}, shared between I and J, and asymmetric keys k_I^+, k_I^- representing corresponding public and private keys belonging to I. Notice that keys are never transmitted on the network, thus modelling long-term encryption keys. We also presuppose a set $\mathcal{T}$ of tags. Terms can be tagged using a $\mathsf{Tag} \in \mathcal{T}$, thus determining their role in the authentication task. Moreover, terms can be composed in pairs or encrypted using keys.[3] The special name ε, which denotes the empty message, will be always omitted, e.g., $\mathsf{begin}_\varepsilon(A,B,M_1;\varepsilon)$ will be written as $\mathsf{begin}(A,B,M_1;)$.

Processes (or *protocols*), ranged over by P and Q, behave as follows: $\mathsf{new}(n).P$ generates a fresh name n local to P. We presuppose a unique anonymous public channel, the network, from/to which all principals, including intruders, read and send messages. Similarly to *Lysa*, our input primitive may atomically test part of the read message, by employing pattern-matching. If the input message matches the input term, then the variables occurring in the term are bound to the remaining sub-part of the message; otherwise the message is not read at all. This mechanism is also used to decrypt received messages on-the-fly, and thus constitutes an important novelty compared to the ρ-spi calculus; of course, in order to immediately match a message encrypted with asymmetric cryptography, the correct decryption key has to be specified in the term. For example, process '$\mathsf{in}(\{\!|\, x \,|\!\}_{k_A^-}).P$' reads any message encrypted with A's public key k_A^+, i.e., messages of the form $\{G\}_{k_A^+}$, decrypts them on the fly, and binds all the free occurrences of x to G in process P. The semantics is formally defined in Section 2.2. Note that we distinguish between the *static* cryptographic terms C of the calculus, and the actual sent and received *messages* G. Encryption for terms is denoted $\{\!|\, C \,|\!\}_k$ while encrypted messages are denoted $\{G\}_k$. This is crucial in the calculus semantics to distinguish between the simple reception and the reception-with-decryption of an encrypted message. For instance, $\mathsf{in}(x).P$ and $\mathsf{in}(\{\!|\, y \,|\!\}_{k_{AB}}).Q$ can both read message $\{G\}_{k_{AB}}$ but the first process just reads it, denoted $in(\{G\}_{k_{AB}})$, while the second one reads and decrypts it, denoted $in(\{\!|\, G \,|\!\}_{k_{AB}})$. In particular, x is bound to $\{G\}_{k_{AB}}$ while y is bound to G. Using variables to represent received values whose value is unknown to the recipient goes back to the work on symbolic model checking and constraint solving [5, 11, 25].

[3] For the sake of readability, in the rest of the paper we omit brackets: for instance, the nested pair $((a,b),k)$ is simplified in a,b,k.

The $\mathsf{begin}_N(A,B,M_1;M_2)$ and $\mathsf{end}_N(B,A,M_1;M_2)$ primitives are used to check the *correspondence assertions* [26] in a nonce handshake between A and B based on nonce N. The former primitive declares that A is starting a protocol session with B, while the latter declares that B is ending a protocol session in which he believes to have correctly authenticated A. Messages M_1 and M_2, when specified, represent messages exchanged respectively from B to A and from A to B during the protocol session. Finally, $A \triangleright P$ represents principal A executing process P; process $P|Q$ is the parallel composition of P and Q; process $!P$ indicates an arbitrary number of parallel instances of P, and $\mathbf{0}$ is the null process that does nothing. We often omit $\mathbf{0}$ from protocol specifications. Finally, we remark that the ρ-spi calculus comes with a notion of well-formedness checking that (i) identity declarations do not nest; (ii) the first identity in begin and end assertions refers to the principal running the process; and (iii) principals only use their own keys. We shall always consider well-formed processes.

Example 1. To illustrate, let us consider the following simple challenge-response protocol, where B challenges A to encrypt a freshly generated nonce n, together with the identifier B and a message m, using a shared key. The idea of the protocol is that only A, who shares the long term key k_{AB} with B, is able to send the correct reply. The nonce n ensures that the reply is not a replication of an old message from A; the identity label B is used for specifying the intended receiver of the ciphertext, thus preventing reflection attacks, in which the reply generated by A is reflected back to A in a parallel session started by an intruder. The absence of B would make it impossible for A to tell whether such a message has been originated by B or by herself (see, e.g., [24] for more detail).

$$\begin{array}{lcl@{\qquad}l}
A & \leftarrow n \longrightarrow & B & !A \triangleright \mathsf{in}(x).\mathsf{new}(m).\mathsf{begin}_x(A,B;m).\mathsf{out}(\{\!|B,m,x|\!\}_{k_{AB}}) \\
 & \longrightarrow \{B,m,n\}_{k_{AB}} \rightarrow & & |\ !B \triangleright \mathsf{new}(n).\mathsf{out}(n).\mathsf{in}(\{\!|B,y,n|\!\}_{k_{AB}}).\mathsf{end}_n(B,A;y)
\end{array}$$

Notice the use of variables x and y to receive the nonce and the message, respectively. Recall that names are checked for equality, when specified in a term. So, for example, $\mathsf{in}(\{\!|B,y,n|\!\}_{k_{AB}})$ receives and decrypts the message only if the key is k_{AB} and the first and third components are B and n, respectively. Through $\mathsf{begin}_x(A,B;m)$, A declares the beginning of the protocol with nonce x for authenticating m. On the other side, B ends up the protocol after checking that n is encrypted with k_{AB} and, through $\mathsf{end}_n(B,A;y)$, he declares to have authenticated "A sending message y to B" by nonce n. □

2.2 Operational Semantics of our Dialect of the ρ-spi Calculus

We define the operational semantics of our dialect of ρ-spi in terms of *traces*, following [11]. A trace is a possible sequence of *actions* performed by a process. Each process primitive has an associated action and we denote with *Act* the set of all possible actions. The dynamics of the calculus is formalized by means of a transition relation between *configurations*, i.e., pairs $\langle s,P\rangle$, where $s \in Act^+$ is a trace and P is a closed process. Each transition $\langle s,P\rangle \rightarrow \langle s :: \alpha, P'\rangle$ simulates one computation step in P and records the corresponding action α in the trace. We denote by $\rightarrow^+$ a finite non-empty sequence of computation steps. Principals do not directly synchronize with each other. Instead, they may receive from the unique channel an arbitrary message known to the environment, which models the Dolev-Yao intruder [15]: the knowledge of the environment is formalized by a set of deduction rules stating that the environment knows all the messages sent on the network, every name which is not restricted in the trace, all public keys, and every shared and private key with E in the subscript, e.g., k_{EI} and k_E^-. Moreover, the environment can tag and untag messages, construct or destruct pairs, and encrypt/decrypt messages if the appropriate key is known.

Definition 1 (Traces). *The set $T(P)$ of traces of P is the set of all the traces generated by a finite sequence of transitions from $\langle \varepsilon, P\rangle$: $T(P) = \{s \mid \exists P'$ s.t. $\langle \varepsilon, P\rangle \rightarrow^+ \langle s, P'\rangle\}$*

The notion of safety extends the *correspondence* property of [26, 21] by distinguishing between received and sent messages and pointing out the nonce used in the handshake.

Definition 2 (Safety). *s is safe if and only if whenever $s = s_1 :: end_n(B,A,G_1;G_2) :: s_2$, there exist s_1', s_1'' such that $s_1 = s_1' :: begin_n(A,B,G_1;G_2) :: s_1''$ and $s_1' :: s_1'' :: s_2$ is safe. P is safe if s is safe for all $s \in T(P)$.*

A trace is safe if every $end_n(B,A,G_1;G_2)$ is preceded by a distinct $begin_n(A,B,G_1;G_2)$. Intuitively, this guarantees that whenever B authenticates A receiving G_1 and sending G_2, then A has received G_1 and has sent G_2 in a handshake with B based on nonce n.

3 CR Calculus

In the previous section, we presented a calculus for specifying and reasoning on cryptographic authentication protocols. Now, we want to abstract away from the specific structure of messages, and in particular from symbolic cryptography, and to reason on authentication protocols just in terms of challenges and responses. To illustrate, let us consider again the protocol of Example 1. Intuitively, the first message is a public challenge and the second one is a private response directed to B, as depicted below:

$$\begin{array}{ccc} A & \leftarrow \mathrm{Chal}_n^{\mathsf{Pub},\mathsf{Priv}}(B,A) - & B \\ & - \mathrm{Resp}_n^{\mathsf{Pub},\mathsf{Priv}}(A,B,m) \rightarrow & \end{array}$$

The idea is that cryptographic challenges or responses are abstracted into two special messages, namely $\mathrm{Chal}_N^{\ell,\ell'}(B,A,M_1)$ and $\mathrm{Resp}_N^{\ell,\ell'}(A,B,M_2)$. The former may be read as "challenge sent by B to A for authenticating message M_1 with nonce N" and the latter may be read as "response sent by A to B for authenticating message M_2 with nonce N". Labels $\ell, \ell' \in \{\mathsf{Pub}, \mathsf{Taint}, \mathsf{Int}, \mathsf{Priv}\}$ specify the security level of the challenge and the response, respectively, with the following meaning:

Pub **:** readable and modifiable by everyone; cleartexts fall in this class;
Taint **:** modifiable by everyone but readable only by the intended receiver; public key cryptography is an example of this kind;
Int **:** readable by everyone but modifiable only by the sender (e.g., digital signatures);
Priv **:** readable only by the intended receiver and modifiable only by the sender; this is achievable through, e.g., “authenticated symmetric encryption” as soon as the “direction” is made explicit as previously explained.

Table 2 Protocol concretizations

A	(protocol a)	B	A	(protocol b)	B	A	(protocol c)	B
	$\leftarrow n$ —			$\leftarrow \{B,n,m_1\}_{k_A^+}$ —			$\leftarrow \{B,m,n\}_{k_{AB}}$ —	
	— $\{B,m,n\}_{k_A^-} \rightarrow$			— $\{A,n,m_2\}_{k_B^+} \rightarrow$			— $\{B,m,n\}_{k_{AB}} \rightarrow$	

Table 3 CR Calculus Names and Terms

$(\mathsf{CR}$ **Names**$)$ $\mathsf{a} ::=$	n,m	(Msgs)	$(\mathsf{CR}$ **Terms**$)$ $\mathsf{C} ::=$	a	(Names)
	$\mathsf{I},\mathsf{J},\mathsf{A},\mathsf{B},\mathsf{E}$	(Identities)		$\mathsf{x},\mathsf{y},\mathsf{z}$	(Vars)
	$\top$	(Any)		$(\mathsf{C}_1,\mathsf{C}_2)$	(Pair)
Notation : $\ell \in \{\mathsf{Pub},\mathsf{Taint},\mathsf{Priv},\mathsf{Int}\}$				$\mathrm{Chal}_{\mathsf{N}}^{(\ell,\ell')}(\mathsf{I},\mathsf{J},\mathsf{M})$	(Chal)
M,N denote terms with no challenges and responses				$\mathrm{Resp}_{\mathsf{N}}^{(\ell',\ell)}(\mathsf{I},\mathsf{J},\mathsf{M})$	(Resp)

In the example above, the labels $\mathsf{Pub},\mathsf{Priv}$ specify that the challenge is public, i.e., in clear, and the response is private, i.e., ciphered with a symmetric key. The capability of reading or modifying challenges/responses induces a partial order $\leq$ on labels:

$\mathrm{Chal}_n^{(\ell,\ell')}(A,B,m)$
$\mathrm{Resp}_n^{(\ell',\ell)}(A,B,m)$

$$\begin{tikzcd}
 & \mathsf{Priv}(B\ reads, A\ \ modifies) \arrow[dl, dash] \arrow[dr, dash] & \\
\mathsf{Taint}(B\ reads,\ all\ modify) \arrow[dr, dash] & & \mathsf{Int}(all\ read, A\ modifies) \arrow[dl, dash] \\
 & \mathsf{Pub}(all\ read,\ all\ modify) &
\end{tikzcd}$$

Messages with $\ell \leq \mathsf{Taint}$ may be modified by everyone, while messages with $\ell \leq \mathsf{Int}$ may be read by everyone. Moreover, only B can read messages with $\ell \geq \mathsf{Taint}$ and only A can generate messages with $\ell \geq \mathsf{Int}$. There are two ways in which a principal B can authenticate himself: (i) by sending a response that only B can generate, i.e., Int or Priv ; and (ii) by replying to a challenge that only B can read, i.e., Taint or Priv .

For example, $\mathrm{Resp}_n^{\mathsf{Pub},\mathsf{Int}}(A,B,m)$ represents an integer response to a public challenge $\mathrm{Chal}_n^{\mathsf{Pub},\mathsf{Int}}(B,A)$, which might be concretized through symbolic cryptography as in protocol a (cf. Table 2): A proves her identity by signing the nonce together with 'B' and message m. As another example, consider $\mathrm{Resp}_n^{\mathsf{Taint},\mathsf{Taint}}(A,B,m_2)$ representing a tainted response to a tainted challenge $\mathrm{Chal}_n^{\mathsf{Taint},\mathsf{Taint}}(B,A,m_1)$, which might be concretized as in protocol b: A proves her identity by decrypting the tainted challenge using her private key. Attention should be payed when the security label of the challenge and the response is the same and symmetric key cryptography is used. For example, $\mathrm{Resp}_n^{\mathsf{Priv},\mathsf{Priv}}(A,B,m_2)$ with challenge $\mathrm{Chal}_n^{\mathsf{Priv},\mathsf{Priv}}(B,A,m_1)$ should never be concretized using messages that might be confused. For example, consider the worst case in which challenge and response are the same, as in protocol c. Here the enemy can trivially attack the protocol by replaying the received challenge back to B, authenticating as A. This is why the CR calculus explicitly distinguishes between challenges and responses and the abstraction from ρ-spi to the CR calculus guarantees that they remain distinguishable even when concretized through symbolic cryptography (see Section 4).

3.1 Syntax and semantics

The calculus of Challenges and Responses, called CR calculus, has the same syntax as our dialect of ρ-spi calculus apart from names and terms, reported in Table 3. CR names, noted a, correspond to the names in ρ-spi plus $\top$, used for abstracting an arbitrary ρ-spi term. Terms, instead, have neither tags nor symbolic cryptography, but include challenges and responses. CR processes, ranged over by P, have the same syntax as ρ-spi processes but use the CR names and terms described above.

Example 2. We show the CR calculus specification corresponding to Example 1.

$$\begin{array}{l} !\mathsf{A} \triangleright \mathsf{in}(\mathsf{Chal}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{B},\mathsf{A})).\mathsf{new}(\mathsf{m}).\mathsf{begin}_{\mathsf{x}}(\mathsf{A},\mathsf{B};\mathsf{m}).\mathsf{out}(\mathsf{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{B},\mathsf{m})) \\ \mid\ !\mathsf{B} \triangleright \mathsf{new}(\mathsf{n}).\mathsf{out}(\mathsf{Chal}_{\mathsf{n}}^{\mathsf{Pub,Priv}}(\mathsf{B},\mathsf{A})).\mathsf{in}(\mathsf{Resp}_{\mathsf{n}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{B},\mathsf{y})).\mathsf{end}_{\mathsf{n}}(\mathsf{B},\mathsf{A};\mathsf{y}) \end{array}$$

A reads a public challenge on nonce *x* coming (apparently) from *B*. She generates a new message *m*, starts the protocol session, and sends to *B* a private reply containing *m* and based on the received nonce *x*. On the other side, *B* generates the nonce *n*, sends the public challenge to *A*, waits for the private reply containing the same nonce *n*, and commits. Authentication of *A* is guaranteed by the fact that only *A* can generate such a response and the nonce is fresh and used only once. □

As for the ρ-spi calculus, we have a notion of process well-formedness that rules out undesired process behaviors and enforces the scope of challenges and responses to the principals that can actually generate and receive them. In the following, we shall always consider well-formed processes. For more detail, we refer to the long version [6].

We define CR calculus operational semantics in terms of *traces*. Recall that principals do not directly synchronize with each other. Instead, they may receive from the unique channel an arbitrary message known to the environment, which models the Dolev-Yao intruder. Here, instead of symbolic representations of cryptographic terms, the intruder can manipulate challenges and responses but we assume that security labels are respected: the intruder can forge messages with labels less than or equal to Taint; moreover, the intruder can read messages with labels less than or equal to Int. In the following, we write $\rightarrow_{\mathsf{a}}$ and $\rightarrow_{\mathsf{a}}^{+}$ to denote one-step and multi-step reduction of abstract processes, respectively.

4 Abstraction from ρ-spi to CR

In this section, we define the abstraction of protocol narrations from ρ-spi into CR calculus. The idea is to abstract away from the specific structure of messages, focusing instead on their challenge-response role in the authentication task. The abstraction is fully automated and, given a ρ-spi protocol narration, yields a CR protocol that is proved to be a faithful abstraction of the former.

4.1 Abstraction Function

The abstraction is defined on ρ-spi terms, traces and processes. The abstraction of terms, reported in Table 4, allows for abstracting ciphertexts into challenge and response components: the abstraction of traces and processes is conceptually simpler and amounts to

Table 4 Abstraction

Terms

$$\begin{aligned}
\alpha(a) &= \mathsf{a}\\
\alpha(x) &= \mathsf{x}\\
\alpha(C_1,C_2) &= (\alpha(C_1),\alpha(C_2))\\
\alpha(\mathsf{Tag}(C)) &= \alpha(C)\\
\alpha(\{\!|C|\!\}_k) &= \begin{cases} \underline{f}(\{\!|C|\!\}_k) & \text{if } \{\!|C|\!\}_k \in dom(\underline{f})\\ \alpha(C) & \text{otherwise}\end{cases}
\end{aligned}$$

where:

- $f : \{\!|C|\!\}_k \mapsto \mathrm{Chal/Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})$
- $\forall C \in dom(f), \sigma : \mathit{Vars} \to \mathit{Names} \cup \mathit{Ciphertexts}$,
$\underline{f}(C\sigma) = f(C)\sigma^{\natural}$, where $\sigma^{\natural}$ is defined as follows:
  - (i) $\sigma^{\natural} : \mathit{Abstract\ Vars} \to \mathit{Abstract\ Terms}$
  - (ii) $\mathsf{x} \in dom(\sigma^{\natural}) \Leftrightarrow x \in dom(\sigma)$
  - (iii) $\sigma^{\natural}(\mathsf{x}) = \begin{cases} \mathsf{a} & \text{if } \sigma(x) = a\\ \top & \text{otherwise}\end{cases}$

Table 5 Encryption Abstraction

A partial function $f : \{\!|C|\!\}_k \mapsto \mathrm{Chal/Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})$ is an *encryption abstraction* iff

Format - $names(C) \subseteq \mathcal{ID}$ and $names(\mathsf{N}) = names(\mathsf{M}) = \emptyset$;
- $x \in vars(C)$ if and only if $\mathsf{x} \in vars(\mathsf{N}) \cup vars(\mathsf{M})$;
- C does not contain encryptions

Unique Abstraction if $\exists C, C' \in dom(f), \sigma, \sigma' : \mathit{Vars} \to \mathit{Names} \cup \mathit{Ciphertexts}$ s.t. $C\sigma = C'\sigma'$, then $C = C'$ (the closure $\underline{f}$ of f is a partial function);

Encryption If $\{\!|C|\!\}_k \in dom(f)$ then

– $k = k_I^{+}$ implies $f(\{\!|C|\!\}_k) \in \{\mathrm{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{J},\mathsf{I},\mathsf{M}), \mathrm{Resp}_{\mathsf{N}}^{\ell',\ell}(\mathsf{J},\mathsf{I},\mathsf{M})\}$, with $l \leq \mathsf{Taint}$;
– $k = k_I^{-}$ implies $f(\{\!|C|\!\}_k) \in \{\mathrm{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M}), \mathrm{Resp}_{\mathsf{N}}^{\ell',\ell}(\mathsf{I},\mathsf{J},\mathsf{M})\}$, with $l \leq \mathsf{Int}$;
– $k = k_{IJ}$ implies $f(\{\!|C|\!\}_k) \in \{\mathrm{Chal/Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M}), \mathrm{Chal/Resp}_{\mathsf{N}}^{\ell',\ell}(\mathsf{J},\mathsf{I},\mathsf{M})\}$;

abstract every term occurring therein. The abstraction of names and variables is straightforward and amounts to map them to their corresponding CR counterparts. The abstraction of a pair yields the pair composed of the abstractions of each component. Finally, tags are abstracted away. The abstraction of encryptions performed by trusted principals is parameterized and ruled by a partial function f from encryptions to abstract challenge-response terms. This function works on syntactic terms. At run time, variables may be replaced by ground terms: the abstraction of the resulting encryption is ruled by the closure of f, denoted by $\underline{f}$, which behaves as f apart from abstract variables that are replaced by either abstract names or $\top$, depending on whether the corresponding concrete variable is instantiated to a name or a different term, i.e., a ciphertext. This simulates the semantics of the CR calculus, which instantiates variables only to abstract names. Instead of fixing function f, we let the programmer define, and extend when needed, such a function. The abstraction is proved to be safe as far as f is an *encryption abstraction* , namely it respects the conditions reported in Table 5 and discussed below:

Format The only names occurring in C are identities; N and M are abstract terms only composed of the (abstraction of) variables occurring in C. It is important that abstractions preserve all of the encrypted variables so that, when decryption is performed, we can recover those (abstract) values from the corresponding challenge-response. For the sake of simplicity, here we do not consider nested encryptions: this issue is addressed in the long version of this paper [6].

Unique Abstraction We require that, when f is closed by substitution of variables to names and ciphertexts, it remains a function. This means that given a ground run-time message G, it has a unique possible abstraction $\underline{f}(G)$; without this constraint, abstraction α in Table 4 would not be a function. This condition may be easily verified by applying a standard most general unifier.

Encryption We require that encrypted messages are abstracted at most at the same security level of the concrete messages. Public key encryption is at most Taint, digital signature is at most Int and symmetric key encryption has no constraints as it is at the highest level (Priv). Identities are placed in the correct positions: a public key specifies the intended receiver, a signature proves the identity of the sender and a symmetric key can be only used by the respective owners. Notice that it is allowed to abstract a message to a lower level of security. This is not a problem for the soundness of the abstraction but could make the abstract protocol insecure even if the concrete one is safe, thus losing precision in the abstraction.

Note that we identify bindings between encryptions and abstract messages up to capture-avoiding substitution of names and variables, including the ones of key subscripts. (e.g., $f(\{\!|A,x,z|\!\}_{k_{AB}}) = \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{B},\mathsf{z})$ implies $f(\{\!|A,x,z|\!\}_{k_{AC}}) = \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{C},\mathsf{z})$.)

Example 3. To illustrate, let us consider again the protocol of Example 1. Let us define the encryption abstraction as $f = \{\!|B,z,x|\!\}_{k_{AB}} \mapsto \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{B},\mathsf{z})$. The abstract narration resulting from the protocol abstraction is reported below.

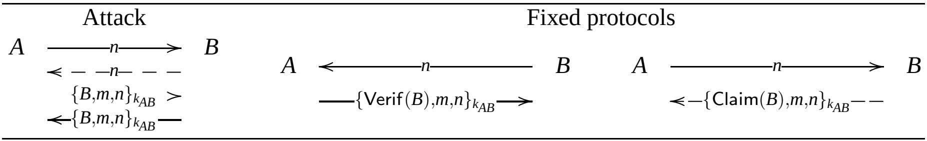
$$\begin{array}{l} !\mathsf{A} \triangleright \mathsf{in(x).new(m).begin_x(A,B;m).out(Resp_x^{Pub,Priv}(A,B,m))} \\ \mid\, !\mathsf{B} \triangleright \mathsf{new(n).out(n).in(Resp_n^{Pub,Priv}(A,B,y)).end_n(B,A;y)} \end{array}$$

This process is the same as the one in Example 2 apart from the nonce sent in clear that is abstracted in n instead of $\mathrm{Chal}_{\mathsf{n}}^{\mathsf{Pub,Priv}}(\mathsf{B},\mathsf{A})$. Indeed, the nonce does not contain enough information for determining whether it is a challenge or a response. This issue is addressed in Section 5.1. Suppose now that we want to abstract a second protocol, similar to the previous one but with A instead of B as identity label inside the ciphertext. The protocol and the abstraction function for the ciphertext are reported below:

$$\begin{array}{c} \text{---}\, n \rightarrow \\ \leftarrow \{B,m,n\}_{k_{AB}} \text{---} \end{array} \qquad\qquad g = \{\!|B,z,x|\!\}_{k_{AB}} \mapsto \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{B},\mathsf{A},\mathsf{z})$$

Unfortunately, f and g cannot be combined together as they map the same encryption to two different abstract messages, since $f(\{\!|B,z,x|\!\}_{k_{AB}}) = \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{B},\mathsf{z})$ and $g(\{\!|B,z,x|\!\}_{k_{AB}}) = \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{B},\mathsf{A},\mathsf{z})$, thus violating property *Unique Abstraction* in Table 5. As a matter of fact, the two protocols are safe as far as they are not put in parallel composition. Indeed, their concurrent execution gives rise to the standard *reflection* attack depicted on the left side of Table 6. A runs the first protocol ($\rightarrow$) as responder and the second one ($\dashrightarrow$) as initiator: at the end, she authenticates B even if B is not present at all in the authentication session. The problem is that A generates in the first protocol a ciphertext that the enemy may reply to A, by impersonating B running the second protocol. The problem can be fixed, and the attack prevented, by tags telling the claimant from the verifier, as shown in Table 6. These tags make the ciphertexts syntactically different and, consequently, the union of the two "fixed" functions $f_{fix} = \{\mathsf{Verif}(B),z,x\}_{k_{AB}} \mapsto \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{A},\mathsf{B},\mathsf{z})$ and $g_{fix} = \{\mathsf{Claim}(B),z,x\}_{k_{AB}} \mapsto \mathrm{Resp}_{\mathsf{x}}^{\mathsf{Pub,Priv}}(\mathsf{B},\mathsf{A},\mathsf{z})$ is still a function and, in particular, an encryption abstraction.

Table 6 Attack on the multi-protocol system and fix


Attack
Fixed protocols
A
B
n
n
$\{B,m,n\}_{k_{AB}}$
$\{B,m,n\}_{k_{AB}}$
A
B
n
$\{\mathsf{Verif}(B),m,n\}_{k_{AB}}$
A
B
n
$\{\mathsf{Claim}(B),m,n\}_{k_{AB}}$

The main result states that every concrete computation has a direct counterpart in the abstract model. Indeed, we abstract away from the structure of terms, while every reduction rule in the concrete semantics has a representative in the abstract one.

Theorem 1 (Soundness). *If* $\langle s,P\rangle \to^{+} \langle s',P'\rangle$*, then* $\langle \alpha(s),\alpha(P)\rangle \to_{\mathsf{a}}^{+} \langle \alpha(s'),\alpha(P')\rangle$.

5 Effect System

In this section, we overview an effect system for determining the safety of CR processes: the full formalization is in Appendix A. Our use of effects for locally checking the correct behavior of principals is inspired from [14]. Superseding [14] however, we do not need any typing environment as CR syntax makes explicit the role of terms, which considerably simplifies the analysis. The effect system checks the safety of nonce handshakes by relying on the following intuitive principles: the *verifier* should authenticate through an end assertion only after the successful completion of a suitable nonce handshake based on a fresh nonce and the *claimant* should respond to a received challenge only after a suitable begin assertion. For instance, A should assert $\mathsf{end}_{\mathsf{n}}(\mathsf{A},\mathsf{I},\mathsf{M}_1;\mathsf{M}_2)$ only after the output of $\mathrm{Chal}_{\mathsf{n}}^{\ell,\ell'}(\mathsf{A},\mathsf{I},\mathsf{M}_1)$ (i.e., a challenge with nonce n for authenticating message M_1), the input of $\mathrm{Resp}_{\mathsf{n}}^{\ell,\ell'}(\mathsf{I},\mathsf{A},\mathsf{M}_2)$ (i.e., a response with the same nonce for authenticating message M_2) and the check on the freshness of n. Dually, B should respond to the challenge $\mathrm{Chal}_{\mathsf{n}}^{\ell,\ell'}(\mathsf{A},\mathsf{I},\mathsf{M}_1)$ through the response $\mathrm{Resp}_{\mathsf{n}}^{\ell,\ell'}(\mathsf{I},\mathsf{A},\mathsf{M}_2)$, only after having asserted $\mathsf{begin}_{\mathsf{n}}(\mathsf{B},\mathsf{J},\mathsf{M}_1;\mathsf{M}_2)$.

5.1 Prevalidation

This reasoning appears conceptually simple and elegant but requires to solve some technical issues: (i) nonces should be used only as subscripts of challenge-response components as they might otherwise be unintentionally leaked; and (ii) nonces sent or received in clear do not contain enough information for predicting whether they represent challenges or responses. We solve both of these problems by relying on a processing of CR protocol descriptions, which is formalized by the function $\mathit{preval} : \mathsf{P} \mapsto \mathsf{P}$ reported in [6]. The idea is to inspect the code for (i) checking that the set of nonces and the set of messages sent inside challenges and responses are disjoint and (ii) replacing every occurrence of nonce N in input or output as "plaintext" (i.e., not in a challenge or response) with either $\mathrm{Chal}_{\mathsf{N}}^{\mathsf{Pub},\ell}(\mathsf{A},\mathsf{I})$ or $\mathrm{Resp}_{\mathsf{N}}^{\ell,\mathsf{Pub}}(\mathsf{A},\mathsf{I})$, depending on whether the term $\mathrm{Resp}_{\mathsf{N}}^{\ell',\ell}(\mathsf{I},\mathsf{A})$ or the term $\mathrm{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{A})$ occurs in the process, respectively. Intuitively, if a nonce N sent in clear is also sent inside an encrypted challenge (resp. response), then we consider such a nonce as a response (resp. challenge). Indeed, a Pub challenge does not force the security level of the response, which might be either Priv or Int. Similarly, a Pub response does not force the security level of the challenge, which might be either Priv or Taint. As a matter of fact, the prevalidation step is harmless since it does not affect the safety of processes.

5.2 Effects and their Usefulness for Proving Safety

The link between correspondence assertions and the nonce handshake followed by principals can be achieved by tracking the generation-reception of challenges and responses and the freshness of nonces. This tracking can be conducted by means of *effects*. Formally, effects are multisets of atomic effects, as formalized by the following grammar:

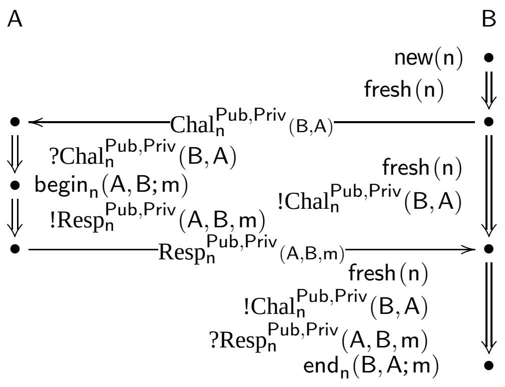
$$\begin{array}{lll} (\textit{at. eff.}) & f & ::= \mathsf{fresh}(\mathsf{n}) \mid [!|?]\mathrm{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M}) \mid [!|?]\mathrm{Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M}) \\ (\textit{eff.}) & e & ::= [f_1,\dots,f_n] \end{array}$$

– The atomic effect $\mathsf{fresh}(\mathsf{n})$ tracks the freshness of nonce n, allowing for a successive $\mathsf{end}_{\mathsf{n}}(\cdots)$ on the same nonce: a new name n is fresh until it is used in an $\mathsf{end}_{\mathsf{n}}(\cdots)$.
– The atomic effect $?\mathsf{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})$ (resp. $!\mathsf{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})$) tracks the reception (resp. generation) of a challenge with nonce N sent by I to J for authenticating M, thus allowing the occurrence of a successive $\mathsf{begin}_{\mathsf{N}}(\mathsf{J},\mathsf{I},\mathsf{M};\cdot)$ (resp. $\mathsf{end}_{\mathsf{N}}(\mathsf{I},\mathsf{J},\mathsf{M};\cdot)$) assertion. The security levels ℓ and ℓ' have the same semantics as in Section 3.
– The atomic effect $?\mathsf{Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})$ tracks the reception of a response with nonce N sent by I to J for authenticating M, thus allowing a successive $\mathsf{end}_{\mathsf{N}}(\mathsf{J},\mathsf{I},\cdot;\mathsf{M})$.
– The atomic effect $!\mathsf{Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})$ has different semantics as it *enables* I to generate a response for J with nonce N for authenticating M. This atomic effect is justified by a $\mathsf{begin}_{\mathsf{N}}(\mathsf{I},\mathsf{J},\cdot;\mathsf{M})$ assertion. This asymmetry is discussed below.

5.3 Effects for the PC handshake

For giving the intuition on the use of effects, let us consider the protocol Example 2. The effects for the process are depicted in Table 7. B generates a nonce n, whose freshness is tracked by the atomic effect $\mathsf{fresh}(\mathsf{n})$, and sends it in a challenge to A: $!\mathrm{Chal}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{B},\mathsf{A})$ tracks on the verifier's code the generation of the challenge. The reception of this message is tracked by $?\mathrm{Chal}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{B},\mathsf{A})$. The following $\mathsf{begin}_{\mathsf{n}}(\mathsf{A},\mathsf{B};\mathsf{m})$ requires the reception of a challenge generated by B (effect $?\mathrm{Chal}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{B},\mathsf{A})$) and justifies the generation of a response for authenticating m (effect $!\mathrm{Resp}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{A},\mathsf{B},\mathsf{m})$). This is the reason why effects of the form $!\mathrm{Resp}_{\mathsf{N}}^{\ell,\ell'}(\dots)$ are used for *enabling* rather than tracking the generation of responses, as mentioned above. Hence $!\mathrm{Resp}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{A},\mathsf{B},\mathsf{m})$ enables A to generate the corresponding response, which is eventually received by B and is tracked by $?\mathrm{Resp}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{A},\mathsf{B},\mathsf{m})$. Since the handshake has been completed (effects $!\mathrm{Chal}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{B},\mathsf{A})$ and $?\mathrm{Resp}_{\mathsf{n}}^{\mathsf{Pub},\mathsf{Priv}}(\mathsf{A},\mathsf{B},\mathsf{m})$) and the nonce is fresh (effect $\mathsf{fresh}(\mathsf{n})$), B can assert $\mathsf{end}_{\mathsf{n}}(\mathsf{B},\mathsf{A};\mathsf{m})$. The main judgement in our analysis is $\vdash \mathsf{P} : e$, read as "P has effect e", meaning that process P is safe under the conditions expressed by effect e. For instance, $\vdash \mathsf{P} : [\mathsf{fresh}(\mathsf{n})]$ means that P is safe if n is fresh.

Table 7 CR protocol with effects


A
B
new(n)
fresh (n)
Chal$_n^{\text{Pub,Priv}}$(B,A)
?Chal$_n^{\text{Pub,Priv}}$(B, A)
fresh (n)
begin$_n$(A, B; m)
!Chal$_n^{\text{Pub,Priv}}$(B, A)
!Resp$_n^{\text{Pub,Priv}}$(A, B, m)
Resp$_n^{\text{Pub,Priv}}$(A,B,m)
fresh (n)
!Chal$_n^{\text{Pub,Priv}}$(B, A)
?Resp$_n^{\text{Pub,Priv}}$(A, B, m)
end$_n$(B, A; m)

5.4 Safety Results

The following theorem states that CR processes with empty effect are safe.

Theorem 2 (Effect Safety). *If* $\vdash \mathsf{P} : []$, *then* P *is safe.*

The following theorem constitutes the main result of our framework: if the abstraction of a process is safe, then such a process is safe as well, i.e., a proof of authentication in the CR calculus automatically entails authentication in the ρ-spi calculus.

Theorem 3 (Safety). *If* $\vdash preval(\alpha(P)) : []$, *then* P *is safe.*

The last theorem states that the effect system is modular and the analysis compositional.

Theorem 4 (Modularity and Compositionality). *Let* P *be an abstract process of the form* $!\mathsf{P}_1 | \dots | !\mathsf{P}_\mathsf{m}$. *Then* $\vdash \mathsf{P} : []$ *if and only if* $\vdash !\mathsf{P}_\mathsf{i} : [], \forall i \in [1, m]$.

Depending on whether one reads P as a protocol, executed by different principals, or a multi-protocol system, made of different protocols, this theorem says that (i) a protocol is safe if all participants are successfully, and independently, checked (*modularity*), and (ii) a multi-protocol system composed of safe protocols is safe (*compositionality*).

6 Conclusion

Finally, we tested our technique on several existing protocols; some examples are in the long version [6], which also contains the extension of the analysis to nested encryptions and mutual authentication protocols, where a ciphertext may be both a challenge and a response. As future work, we plan to exploit recent results on linking symbolic cryptography with actual cryptographic algorithms to verify truly abstract authentication protocols in a way that ensures strong authentication guarantees even for concrete, cryptographic implementations. We also plan to relax the constraint that keys do not circulate on the network, thus extending the scope of our analysis to protocols for session-key distribution, where a session key is exchanged between principals and then possibly used for authenticating.

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-Verlag, 2001.
3. M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. In *Proc. 29th Symposium on Principles of Programming Languages (POPL)*, pages 33–44. ACM Press, 2002.
4. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 36–47, 1997.
5. R. M. Amadio, D. Lugiez, and V. Vanackére. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2003.
6. M. Backes, T. Cortesi, R. Focardi, and M. Maffei. A calculus of challenges and responses (full version). http://www.dsi.unive.it/∼maffei/crcalc.ps.
7. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE Computer Society Press, 2001.
8. B. Blanchet. From secrecy to authenticity in security protocols. In *Proc. 9th International Static Analysis Symposium (SAS)*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359. Springer-Verlag, 2002.
9. C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. Nielson. Automatic validation of protocol narration. In *Proc. 16th IEEE Computer Security Foundations Workshop (CSFW)*, pages 126–140. IEEE Computer Society Press, 2003.
10. 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.
11. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *28rd International Colloquium on Automata, Languages and Programming (ICALP 2001)*, Lecture Notes in Computer Science, pages 667–681. Springer-Verlag, 2001.
12. M. Bugliesi, R. Focardi, and M. Maffei. Principles for entity authentication. In *Proceedings of 5th International Conference Perspectives of System Informatics (PSI 2003)*, volume 2890 of *Lecture Notes in Computer Science*, pages 294–307. Springer-Verlag, July 2003.
13. M. Bugliesi, R. Focardi, and M. Maffei. Compositional analysis of authentication protocols. In *Proc. 13th European Symposium on Programming (ESOP)*, volume 2986 of *Lecture Notes in Computer Science*, pages 140–154. Springer-Verlag, 2004.
14. M. Bugliesi, R. Focardi, and M. Maffei. Dynamic types for authentication, 2006. To appear in Journal of Computer Security.
15. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
16. N. Durgin, J. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):677–721, 2004.
17. J. Feret. *Analysis of Mobile Systems by Abstract Interpretation*. PhD thesis, École Polytechnique, 2005.
18. A. D. Gordon and A. Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3):435–484, 2004.
19. J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 24–34. IEEE Computer Society Press, 2000.

Table 8 Effect System (Terms and Processes)

CHAL
$$\frac{\ell_C \geq \mathsf{Taint} \vee \ell_R \geq \mathsf{Int}}{\vdash \mathsf{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M}) : ([\mathsf{Chal}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})];[])}$$

RESP
$$\frac{\ell_C \geq \mathsf{Taint} \vee \ell_R \geq \mathsf{Int}}{\vdash \mathsf{Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M}) : ([];[\mathsf{Resp}_{\mathsf{N}}^{\ell,\ell'}(\mathsf{I},\mathsf{J},\mathsf{M})])}$$

ATOM
$$\vdash \mathsf{a} : ([];[])$$

PAIR
$$\frac{\vdash \mathsf{C} : (e_C;e_R) \qquad \vdash \mathsf{C}' : (e'_C;e'_R)}{\vdash (\mathsf{C},\mathsf{C}') : (e_C + e'_C; e_R + e'_R)}$$

NIL
$$\vdash \mathbf{0} : []$$

REPL
$$\frac{\vdash \mathsf{P} : []}{\vdash !\mathsf{P} : []}$$

PAR
$$\frac{\vdash \mathsf{P} : e_P \qquad \vdash \mathsf{Q} : e_Q}{\vdash \mathsf{P}|\mathsf{Q} : e_P + e_Q}$$

ID
$$\frac{\vdash \mathsf{P} : e}{\vdash \mathsf{A} \triangleright \mathsf{P} : e}$$

NEW
$$\frac{\vdash \mathsf{P} : e}{\vdash \mathsf{new}(\mathsf{n}).\mathsf{P} : e - \mathsf{fresh}(\mathsf{n})}$$

IN
$$\frac{\vdash \mathsf{P} : e + ?e_C + ?e_R \qquad \vdash \mathsf{C} : (e_C;e_R)}{\vdash \mathsf{in}(\mathsf{C}).\mathsf{P} : e}$$

OUT
$$\frac{\vdash \mathsf{P} : e + !e_C \qquad \vdash \mathsf{C} : (e_C;e_R)}{\vdash \mathsf{out}(\mathsf{C}).\mathsf{P} : e + !e_R}$$

BEGIN
$$\frac{\vdash \mathsf{P} : e + [!\mathsf{Resp}_{\mathsf{N}}^{\ell_C,\ell_R}(\mathsf{A},\mathsf{I},\mathsf{M}_2)] \qquad \mathsf{M}_2 \textit{ ground}}{\vdash \mathsf{begin}_{\mathsf{N}}(\mathsf{A},\mathsf{I},\mathsf{M}_1;\mathsf{M}_2).\mathsf{P} : e + [?\mathsf{Chal}_{\mathsf{N}}^{\ell_C,\ell_R}(\mathsf{I},\mathsf{A},\mathsf{M}_1)]}$$

END
$$\frac{\vdash \mathsf{P} : e \qquad \mathsf{M}_1 \textit{ ground}}{\vdash \mathsf{end}_{\mathsf{n}}(\mathsf{A},\mathsf{I},\mathsf{M}_1;\mathsf{M}_2).\mathsf{P} : e + [!\mathsf{Chal}_{\mathsf{n}}^{\ell_C,\ell_R}(\mathsf{A},\mathsf{I},\mathsf{M}_1), ?\mathsf{Resp}_{\mathsf{n}}^{\ell_C,\ell_R}(\mathsf{I},\mathsf{A},\mathsf{M}_2), \mathsf{fresh}(\mathsf{n})]}$$

An abstract term is ground if it contains neither variables nor $\top$

20. J. D. Guttman and F. J. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, 2002.
21. G. Lowe. “A Hierarchy of Authentication Specification”. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE Computer Society Press, 1997.
22. M. Maffei. *Dynamic Typing for Security Protocols*. PhD thesis, UniversitÃă Ca’ Foscari di Venezia, Dipartimento di Informatica, 2006.
23. C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proc. 2000 DARPA Information Survivability Conference and Exposition (DISCEX)*, pages 237–250, 2000.
24. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
25. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 166–175, New York, NY, USA, 2001. ACM Press.
26. T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operation Systems Review*, 28(3):24–37, 1994.

A Effect system

For simplifying the presentation of processes, in the following we introduce some additional conventions. We write $![f_1,\ldots,f_n]$ and $?[f_1,\ldots,f_n]$ to denote $[!f_1,\ldots,!f_n]$ and

$[?f_1,\dots,?f_n]$, respectively. Furthermore, $+$ and $-$ are the usual union and subtraction operators on multi-sets: e_1+e_2 yields the effect composed of all the atomic effects in e_1 plus the ones in e_2, while e_1-e_2 yields the effect obtained by removing, if present, an occurrence of each atomic effect in e_2 from e_1. If an atomic effect of e_2 does not occur in e_1 then the subtraction of that atomic effect leaves e_1 unchanged.

The binding between abstract messages and their effects is formalized in Table 8 by the judgement $\vdash \mathsf{M} : (e_C; e_R)$, read as "$\mathsf{M}$ has challenge effect e_C and response effect e_R". Notice that we check that the security levels of challenges and responses are consistent, namely either the challenge security level is greater that Taint or the response security level is greater than Int, as discussed in Section 3. The main judgement in our analysis is $\vdash \mathsf{P} : e$, read as "P has effect e", meaning that process P is safe under the conditions expressed by effect e. For instance, $\vdash \mathsf{P} : [\mathsf{fresh}(\mathsf{n})]$ means that P is safe if n is fresh. In the following we shall give informal explanations on the process judgements of Table 8. The validation of a process is defined by induction on its structure and the null process is the base case. NIL validates process **0** under empty effect since the null process is always safe. REPL validates the replication of a process under empty effect [], if that process is in turn validated under empty effect. Requiring the empty effect is necessary in order to preserve the safety; e.g., replicating a process with effect $\mathsf{fresh}(\mathsf{n})$ may generate an infinite number of processes exploiting the freshness of the same nonce n to complete authentication sessions; this, of course, is not safe as a nonce should be used only once. PAR validates the parallel composition of two processes under the union of their effects, stating that the parallel composition of two processes is safe if both of the processes are safe. ID skips identity declarations as they are not relevant in the analysis. NEW justifies, through the atomic effect $\mathsf{fresh}(\mathsf{n})$, at most one use of n as fresh nonce in the continuation process. Indeed, the deletion of one occurrence of $\mathsf{fresh}(\mathsf{n})$ in the thesis allows the continuation process to exploit the nonce for asserting one end assertion (cf. rule END). IN justifies in the continuation process the reception of the challenges and responses composing the received message. OUT justifies in the continuation process the reception of the challenges and requires the permission to generate the responses composing the message sent on the network. As discussed in Section 5.3, the assertion $\mathsf{begin}_{\mathsf{N}}(\mathsf{A},\mathsf{I},\mathsf{M}_1;\mathsf{M}_2)$ (rule BEGIN) requires the reception of $\mathrm{Chal}_{\mathsf{N}}^{\ell_C,\ell_R}(\mathsf{I},\mathsf{A},\mathsf{M}_1)$ and justifies the generation of $\mathrm{Resp}_{\mathsf{N}}^{\ell_C,\ell_R}(\mathsf{A},\mathsf{I},\mathsf{M}_2)$. Similarly, the assertion $\mathsf{end}_{\mathsf{n}}(\mathsf{A},\mathsf{I},\mathsf{M}_1;\mathsf{M}_2)$ (rule END) requires the freshness of n, the generation of $\mathrm{Chal}_{\mathsf{n}}^{\ell_C,\ell_R}(\mathsf{A},\mathsf{I},\mathsf{M}_1)$ and the reception of $\mathrm{Resp}_{\mathsf{n}}^{\ell_C,\ell_R}(\mathsf{I},\mathsf{A},\mathsf{M}_2)$. Notice that we check on the syntax of begin and end assertions that messages sent in the response by the claimant are ground and so are the ones sent in the challenge by the verifier. This suffices for proving that $\top$ never occurs within authenticated messages at run time. This is crucial for carrying safety properties from the abstract semantics to the concrete one, since $\top$ might otherwise be instantiated differently in the matching begin and end assertions. Finally, notice that validation rules univocally determine the process effect from the one of the continuation process, with the exception of END which has to guess the security level of challenges and responses. However, this nondeterminism can be easily solved in the prevalidation by labelling end assertions with the security level of challenges and responses based on the same nonce, thus making the analysis fully deterministic.