Static Validation of Security Protocols*

Chiara Bodei[1], Mikael Buchholtz[2], Pierpaolo Degano[1],
Flemming Nielson[2], Hanne Riis Nielson[2]

1 Dipartimento di Informatica, Università di Pisa
Via F. Buonarroti 2, I-56127 Pisa, Italy. — {chiara,degano}@di.unipi.it

2 Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads bldg 321, DK-2800 Kongens Lyngby, Denmark
— {mib,nielson,riis}@imm.dtu.dk

Draft of March 8, 2004
Please do not distribute

Abstract

We perform a systematic expansion of protocol narrations into terms of a process algebra in order to make precise some of the detailed checks that need to be made in a protocol. We then apply static analysis technology to develop an automatic validation procedure for protocols. Finally, we demonstrate that these techniques suffice for identifying a number of authentication flaws in symmetric and asymmetric key protocols such as Needham-Schroeder symmetric key, Otway-Rees, Yahalom, Andrew Secure RPC, Needham-Schroeder asymmetric key, and Beller-Chang-Yacobi MSR.

1 Introduction

Motivation. Security is a growing concern in the development of software utilising the internet and supporting mobility. An important aspect is to ensure the security of the protocols used for communication: that they do guarantee the necessary amount of confidentiality, authenticity, message integrity, and availability.

Protocol analysis is a hard problem for several reasons. One is that often it is difficult to make precise what are the properties one expects from the protocols; indeed there are examples of protocols that were considered to be secure for many years and then had their security compromised by a slight change in the expectations and a brilliant idea for how to exploit it. Another reason is that

*Supported in part by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies, under the IST-2001-32072 project DEGAS and IST-1999-29075 project SecSafe; the Danish SNF-projects SecSaf and LoST; and the Italian MIUR-project MEFISTO. Parts of this paper appeared in [11].

protocols are often described somewhat informally using *protocol narrations* that are imprecise about some of the finer details concerning the deployment of the protocol. A third reason is that one should guard against all possible kinds of misuse, and it is not easy to give a finitary account of the infinitely many environments in which the protocol will be used.

In practice the problem is even worse. While an implementation of a flawed protocol is likely to lead to an insecure system one should not take it for granted that the implementation of a correct protocol will lead to a secure system. There are far too many potential errors that a programmer can make, as a result of a lack of understanding of the ingredients that actually make the protocol secure, and as a result of sloppy programming; the infamous buffer overflow is a good example of the latter. While our techniques show promise of dealing also with these issues we shall not consider them any further in this paper.

Overview of contribution. We base ourselves on standard *protocol narrations* and extend them (in Section 2) with *annotations* that make it clear how to deal with some of the tests that need to be performed and how to express the authentication intentions of the protocol. We then systematically translate annotated protocol narrations into terms of the process algebra LySA (introduced in Section 3). This is done in such a way that the precision of the extended protocol narration is left unchanged and the intentions can be enforced by a reference monitor that aborts undesired executions. To keep the presentation simple, we consider first a symmetric encryption schema and we then incrementally extend the calculus with asymmetric keys [33].

Next we develop (in Section 4) a *static analysis* for tracking the set of encrypted messages that are successfully being decrypted at each relevant point. We show the semantic correctness of the analysis and demonstrate that best solutions (in the manner of principal types for type systems) always exist. In view of the approximative nature of static analysis the detailed formulation of semantic correctness makes it clear that the analysis might describe a too large set of messages but that no successfully decrypted messages are ever left out.

This allows us (in Section 5) to deal with the general problem of how to give a finite account of an infinity of hostile environments. We adapt the classical approach of Dolev and Yao [34] to model the ability of attackers to send and receive messages and to perform encryptions as well as decryptions. We formally show that the approach realises the notion of "hardest attackers" [55] developed for firewall security in Mobile Ambients.

We then address authenticity, and we see it as a definition/use problem in a world that implicitly includes the Dolev-Yao attacker, as sketched above. Indeed, we statically verify whether a message encrypted by principal A and intended for principal B does indeed come from A and reaches B only. So the two share a secret and authenticate each other. This suffices for dealing with authenticity problems in the protocols mentioned above, in particular with our running example, the Wide Mouthed Frog protocol. Because of the approximative nature of static analysis mentioned above, we may well fail to authenticate

a protocol that is in fact correct but we shall never authenticate a protocol that is in fact flawed.

Our analysis is fully automatic and always terminating; we briefly outline (in Section 6) our implementation that runs in polynomial-time in the size of the LySa formulation of the protocol. We follow the approach taken in [56], based on tree grammars, for translating the problem from an infinite universe to a tree grammar problem over a finite universe. Actual implementations are supported by the Succinct Solver [57]; while the specification of the analysis is compositional the actual solving procedure requires the entire program (and clauses for Dolev-Yao) to be present before computing the solution.

We extend (in Section 8) LySa with asymmetric keys, and we show that only small incremental additions are needed to analyse protocols that use this encryption schema. Our approach proves then to be rather flexible; as a matter of fact, our analysis can be further extended with other features typically used in protocol design, see [17].

We demonstrate (in Section 7) that our technique is strong enough to report the known problems in symmetric key protocols such as Needham-Schroeder [52, 53], Otway-Rees [60], Yahalom [18] and Andrew Secure RPC [67]. Also, we analyse asymmetric key protocols, and we discuss the outcome on Needham-Schroeder (public key)[52, 45] and the Beller-Chang-Yacobi MSR [8]. The latter uses a combination of asymmetric and symmetric key cryptography and here our technique proved powerful enough to discover a flaw that, to the best of our knowledge, has not previously been reported in the literature. Furthermore, our technique is also strong enough that it does not report flaws in suitably amended versions of these protocols.

We conclude with a discussion on related work (in Section 9) and with an assessment of our approach and its ability to deal with related security notions (in Section 10).

The Appendix A summarises the protocol narrations considered, and the Appendix B compares LySa with the Spi-calculus. Proofs of theorems and lemmata presented throughout the paper are collected in Appendix C.

2 Expanding Protocol Narrations

In the literature, security protocols are usually described using an informal notation that leaves implicit some assumptions and does not completely state the actions internal to the principals, as discussed in [2].

Parties in a security protocol interact with an uncertain environment, where some of the participants are not fully trusted or maybe hostile. Consequently, even though carefully designed, protocols may have flaws, allowing malicious agents or *attackers* to violate security. An attacker – according to the classical Dolev and Yao model [34] – gaining some control over the communication network, is able to intercept or forge or invent messages to convince agents to reveal sensitive information or to believe it is one of the legitimate agents in the session.

Consider the following version [5] of the Wide Mouthed Frog protocol [18] (abbreviated hereafter WMF) aiming at establishing a secret (symmetric) session key K between the two principals A and B sharing master keys K_A and K_B, respectively with a trusted server S:

$$\begin{array}{lll} 1. & A \rightarrow S: & A, \{B, K\}_{K_A} \\ 2. & S \rightarrow B: & \{A, K\}_{K_B} \\ 3. & A \rightarrow B: & \{m_1, ..., m_k\}_K \end{array}$$

Usually protocols narrations come along with additional assumptions. For the WMF, one has to say that in the first message A sends to S its name, and then a fresh key K and the name of the intended receiver B, encrypted under the key K_A. In the second one, S forwards the key and the initiator name A to B, encrypted under the key K_B. Finally, A sends B a long sequence of messages $m_1, ..., m_k$ encrypted under the session key K.[1]

Bridging the gap between informal and formal specification is the first and crucial step in programming protocols. As seen above, protocol narrations only list the messages to be exchanged, leaving it unspecified which are the actions to be performed in receiving these messages (inputs, decryptions and possible checks on them), e.g. B should use the content of the second message to decrypt the third message. Furthermore, security goals are left implicit.

As a first step, we unfold the protocol narration in the following extended narration, where we distinguish between outputs and the corresponding inputs, and between encryptions and the corresponding decryptions. Also, we are explicit on which keys are fresh and on which checks are to be performed on received values.

Furthermore, messages are enriched with source address as well as destination address (i.e. the intended ones in an honest exchange), as in IP versions 4 and 6. They are passed in clear and are therefore forgeable. Thus, the general form will be: $source, destination, message_1, \cdots, message_k$ followed by assumptions or checks in square brackets:

$$\begin{array}{llll} 1. & A \rightarrow & : & A, S, A, \{B, K\}_{K_A} [\text{assuming } K \text{ is a new key}] \\ 1'. & \rightarrow S & : & x_A, x_S, x'_A, x \qquad [\text{check } x_S = S, x_A = x'_A] \\ 1''. & S & : & \text{decrypt } x \text{ as } \{x_B, x_K\}_{K_{x_A}} \\ 2. & S \rightarrow & : & S, x_B, \{x_A, x_K\}_{K_{x_B}} \\ 2'. & \rightarrow B & : & y_S, y_B, y \qquad [\text{check } y_B = B] \\ 2''. & B & : & \text{decrypt } y \text{ as } \{y_A, y_K\}_{K_B} \\ 3. & A \rightarrow & : & A, B, \{m_1, \cdots, m_k\}_K \\ 3'. & \rightarrow B & : & z_A, z_B, z \qquad [\text{check } z_B = B, z_A = y_A] \\ 3''. & B & : & \text{decrypt } z \text{ as } \{z_1, \cdots, z_k\}_{y_K} \end{array}$$

The first line describes the actions of the sender of the message, together with the assumption of K being a new key while the next two lines describe

[1] When analysing protocols, we set k to be a large constant to reduce the likelihood of flaws due to spurious interactions between the key and the message exchange phases of the protocol.

the actions of the recipient. After each input we check whether or not the input was actually meant for the recipient (the first checks in lines $1'$, $2'$ and $3'$). Additionally, line $1'$ checks the internal consistency of the message and line $3'$ checks that the identity of the sender corresponds to the one found in the second message. Note that the checks are local to the recipient and do not make the assumption that a recipient has a priori knowledge about the sender of the message such as checking that $x_A = A$ in line $1'$.

As a second step, we look for a way of including the specification of the security goals to be verified. This implies a further refinement of our narrations in terms of assertions for specifying properties. Here, we are interested in authentication properties that rely on the fact that sensible information is sent and received by the principals intended by the protocols. That is, a principal would like to ascertain the origin of a message being received, and also the destination of a message being sent. Consequently, we refine the narration by specifying origin and destination of encrypted messages. Back to our example, we will write, e.g., $\{B, K\}_{K_A}[\text{dest } S]$ to say that in line 1 the encrypted value is intended for S only. Correspondingly, the decrypt action of S in line $1''$ will be annotated with [orig x_A].

For the WMF protocol above we obtain the following extended narration:

$$
\begin{array}{lll}
1. & A \rightarrow & : \; A, S, A, \{B, K\}_{K_A}[\text{dest } S] \; [\text{assuming } K \text{ is a new key}] \\
1'. & \rightarrow S & : \; x_A, x_S, x'_A, x \qquad\qquad [\text{check } x_S = S, x_A = x'_A] \\
1''. & S & : \; \text{decrypt } x \;\; \text{as } \{x_B, x_K\}_{K_{x_A}}[\text{orig } x_A] \\
2. & S \rightarrow & : \; S, x_B, \{x_A, x_K\}_{K_{x_B}}[\text{dest } x_B] \\
2'. & \rightarrow B & : \; y_S, y_B, y \qquad\qquad [\text{check } y_B = B] \\
2''. & B & : \; \text{decrypt } y \;\; \text{as } \{y_A, y_K\}_{K_B}[\text{orig } S] \\
3. & A \rightarrow & : \; A, B, \{m_1, \cdots, m_k\}_K[\text{dest } B] \\
3'. & \rightarrow B & : \; z_A, z_B, z \qquad\qquad [\text{check } z_B = B, z_A = y_A] \\
3''. & B & : \; \text{decrypt } z \;\; \text{as } \{z_1, \cdots, z_k\}_{y_K}[\text{orig } z_A]
\end{array}
$$

Assertions are meant to be added for verification purposes by someone with a global view of the intentions of the protocol and are therefore not restricted to only use local information (e.g. we can have the assertion [orig S] in line $2''$). Other properties, e.g. confidentiality or freshness of various data, can be addressed in the same style (see the Conclusion and [17]).

Note that the two steps leading to an extended protocol narration are systematic. This approach is similar to the one taken by Casper [46], CAPSL [31, 24], CVS [35], and AVISS [7]; see Section 9 for a comparison.

3 The LySA-calculus

The considerations of Section 2 motivate defining a new process algebra, LySA. It is based on the π-calculus, but it differs from this and from the Spi-calculus essentially in two aspects. One difference is the absence of channels: LySA assumes to have one global communication medium to which all processes have access. In our view encodings of protocol narrations should *not* use channels

because the privacy offered by channel based communication may give a degree of security not matched by e.g. ethernet based implementations, where any one can eavesdrop or act as an active attacker; indeed, private channels are often used explicitly as if they were cryptographic keys. Of course, private channels are relevant when modelling intranets: extending LySA with channels only requires minor adjustments to our treatment as demonstrated, e.g. in [17]. The second difference of LySA is that the tests associated with input and decryption are naturally expressed using pattern matching. A more detailed comparison with the Spi-calculus can be found in Appendix B.

Syntax. LySA consists of terms and processes; values then correspond to *closed* terms, i.e. terms without free variables. Values are used to code keys, nonces, messages etc. The syntax of terms E is as follows:

$E ::=$	*terms*		
	n		name ($n \in \mathcal{N}$)
	x		variable ($x \in \mathcal{X}$)
	$\{E_1, \cdots, E_k\}_{E_0}$		symmetric encryption ($k \geq 0$)

Here $\mathcal{N}$ and $\mathcal{X}$ denote sets of names and variables, respectively. Encryptions are tuples of terms $E_1, \cdots, E_k$ encrypted under a term E_0 representing a shared key. We adopt an assumption of perfect cryptography.

The syntax of processes P is mostly familiar to the polyadic Spi-calculus [5]:

$P ::=$	*processes*	
	$\mathsf{0}$	nil
	$\langle E_1, \cdots, E_k \rangle. P$	output
	$(E_1, \cdots, E_j; x_{j+1}, \cdots, x_k). P$	input (with matching)
	$P_1 \mid P_2$	parallel composition
	$(\nu\, n)P$	restriction
	$!\, P$	replication
	$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$	symmetric decryption (with matching)

The set of free variables, resp. free names, of a term or a process is written $\mathsf{fv}(\cdot)$, resp. $\mathsf{fn}(\cdot)$, and is defined in the standard way. As usual we omit the trailing $\mathsf{0}$ of processes.

For ease of presentation here we restrict ourselves to a very simple form of patterns on the form $(E_1, \cdots, E_j; x_{j+1}, \cdots, x_k)$, to be matched against a k-tuple of values $(E'_1, \cdots, E'_k)$. The intuition is that the matching succeeds when the first $1 \leq i \leq j$ values E'_i pairwise correspond to the values E_i, and the effect is to bind the remaining $k-j$ values to the variables $x_{j+1}, \cdots, x_k$. Syntactically, this is indicated by using a semi-colon to separate the components where matching is performed from those where only binding takes place. A more flexible choice is explored in [17].

In our calculus we do not have other data constructors than encryption. We could easily add numbers and operations on these as well as other constructors

and destructors but we do not really need to do so in order to model protocol narrations: we can obtain the same effect using encryption and decryption. Taking pairs as an example, and ignoring the annotations, a pair (E, E') can be rendered as $\{E, E'\}_{\mathsf{PAIR}}$, and a selection mechanism such as $\mathsf{split}\ E\ \mathsf{as}\ (x_1, x_2)\ \mathsf{in}\ P$ can the be rendered as $\mathsf{decrypt}\ E\ \mathsf{as}\ \{; x_1, x_2\}_{\mathsf{PAIR}}\ \mathsf{in}\ P$ assuming of course that PAIR is a name used only for this purpose. We can also deal with history-dependent encryption in the style of [13].

Assertions for origin and destination To describe in LySA the intentions of protocols, we decorate their text with labels, called *crypto-points*, and with *assertions* specifying the origin and destination of encrypted messages. Crypto-points ℓ are from some enumerable set $\mathcal{C}$ (disjoint from $\mathcal{N}$ and $\mathcal{X}$) and are mechanically attached to program points where encryption and decryption occur. Syntactically, when we make an encryption, we have a LySA term on the form:

$$\{E_1, \cdots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]$$

where the assertion $[\mathsf{dest}\ \mathcal{L}]$ specifies the intended crypto-points $\mathcal{L} \subseteq \mathcal{C}$ for decryption of the encrypted value.

Similarly, an encryption occurring in a decrypting process is on the form:

$$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E'_1, \cdots, E'_j; x_{j+1}, \cdots, x_k\}^{\ell}_{E'_0}\ [\mathsf{orig}\ \mathcal{L}]\ \mathsf{in}\ P$$

where $[\mathsf{orig}\ \mathcal{L}]$, specifies the encryption points $\mathcal{L} \subseteq \mathcal{C}$ at which E is allowed to have been encrypted.

Notational conventions. We often write $[\mathsf{dest}\ \ell]$ and $[\mathsf{orig}\ \ell]$ instead of the more cumbersome $[\mathsf{dest}\ \{\ell\}]$ and $[\mathsf{orig}\ \{\ell\}]$.

We shall write $\llbracket \cdot \rrbracket$ for a term with all annotations removed; in particular $\llbracket \{E_1, \cdots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}] \rrbracket = \{\llbracket E_1 \rrbracket, \cdots, \llbracket E_k \rrbracket\}_{\llbracket E_0 \rrbracket}$.

To simplify the definition of the control flow analysis in Section 4, we discipline the α-renaming of bound names. We stipulate that for each name n there is a canonical representative $\lfloor n \rfloor$, and we demand that two names are α-convertible only when they have the same canonical name. A similar assumption applies to variables. The function $\lfloor \cdot \rfloor$ is then extended homomorphically to terms: $\lfloor E \rfloor$ is the term where all names and variables are replaced by their canonical versions. Not to further overload our notation, we simply write E for $\lfloor E \rfloor$ when unambiguous. Finally, we assume that the bound names of a process are renamed apart and that they do not clash with the free names; much in the same way variables are assumed to be all distinct.

Semantics. Following the tradition of the π-calculus, we shall give LySA a reduction semantics. We use the standard notion of substitution, $P[E/x]$, and we slightly modify the usual structural congruence to take care of our disciplined treatment of α-conversion.

More precisely, structural congruence, $\equiv$, is defined on processes to be the least congruence satisfying the following conditions.

(Com)
$$\frac{\wedge_{i=1}^{j} \llbracket E_i \rrbracket = \llbracket E'_i \rrbracket}{\langle E_1, \cdots, E_k \rangle.\, P \;\mid\; (E'_1, \cdots, E'_j;\, x_{j+1}, \cdots, x_k).\, Q \rightarrow_{\mathcal{R}} P \;\mid\; Q[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]}$$

(Decr)
$$\frac{\wedge_{i=0}^{j} \llbracket E_i \rrbracket = \llbracket E'_i \rrbracket \quad \wedge \quad \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{\begin{array}{c}\mathsf{decrypt}\ \{E_1, \cdots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]\ \mathsf{as}\ \{E'_1, \cdots, E'_j;\, x_{j+1}, \cdots, x_k\}^{\ell'}_{E'_0}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P \\ \rightarrow_{\mathcal{R}} P[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]\end{array}}$$

(Par)
$$\frac{P \rightarrow_{\mathcal{R}} P'}{P \mid Q \rightarrow_{\mathcal{R}} P' \mid Q}$$

(Res)
$$\frac{P \rightarrow_{\mathcal{R}} P'}{(\nu\, n)P \rightarrow_{\mathcal{R}} (\nu\, n)P'}$$

(Congr)
$$\frac{P \equiv Q \;\wedge\; Q \rightarrow_{\mathcal{R}} Q' \;\wedge\; Q' \equiv P'}{P \rightarrow_{\mathcal{R}} P'}$$

Table 1: Operational semantics, $P \rightarrow_{\mathcal{R}} P'$, parameterised on $\mathcal{R}$.

- $P \equiv Q$ if P and Q are disciplined α-equivalent;
- $(\mathcal{P}/_{\equiv}, |, 0)$ is a commutative monoid;
- $(\nu\, n)0 \equiv 0$,
  $(\nu\, n)(\nu\, n')P \equiv (\nu\, n')(\nu\, n)P$, and
  $(\nu\, n)(P \mid Q) \equiv P \mid (\nu\, n)Q \quad \text{if } n \notin \mathsf{fn}(P)$;
- $!\, P \equiv P \mid !\, P$

The *reduction relation* $\rightarrow_{\mathcal{R}}$ is the least relation on closed processes, i.e. processes with no free variables, that satisfies the rules in Table 1, where we assume to apply our disciplined α-conversion whenever needed.

As far as the semantics is concerned, we consider two variants. One takes advantage of annotations, the other one discards them:

- the *reference monitor semantics*, written $P \rightarrow_{\mathsf{RM}} Q$, takes $\mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) = (\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L})$; thus, decryptions may only occur at crypto-points designated when the corresponding encryptions were made, and vice-versa, otherwise the execution is stopped;
- the *standard semantics*, written $P \rightarrow Q$, takes, by construction, $\mathcal{R}$ to be universally true.

The rule (Com) expresses that an output $\langle E_1, \cdots, E_j, E_{j+1}, \cdots, E_k \rangle.\, P$ is matched by an input $(E'_1, \cdots, E'_j;\, x_{j+1}, \cdots, x_k).Q$ in case the first j elements are pairwise the same. More precisely, we need to compare E_i with all annotations removed with E'_i with all its annotations removed and to express this we use the operation $\llbracket \cdot \rrbracket$. When the matchings are successful each E_i is bound to each x_i.

Similarly, the inference rule (Decr) expresses the result of matching the term $\{E_1, \cdots, E_j, E_{j+1}, \cdots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]$, resulting from an encryption, against the

pattern in decrypt E as $\{E'_1, \cdots, E'_j; x_{j+1}, \cdots, x_k\}^{\ell'}_{E'_0}$ [orig $\mathcal{L}'$] in P, i.e. the pattern occurring in the corresponding decryption. As it was the case for communication the $[\![E_i]\!]$ must equal the corresponding $[\![E'_i]\!]$ for the first j components and additionally the keys must be the same, i.e. $[\![E_0]\!] = [\![E'_0]\!]$ — this models *perfect* symmetric cryptography. When successful, each E_i is bound to each x_i. In the *reference monitor semantics* we ensure that the crypto-point of the encrypted value is acceptable at the decryption (i.e. $\ell \in \mathcal{L}'$) and that the crypto-point of the decryption is acceptable for the encryption (i.e. $\ell' \in \mathcal{L}$). In the *standard semantics* the condition $\mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})$ is universally true and thus can be ignored. The rules (Par), (Res) and (Congr) are standard.

As noted above, both semantics can be easily extended to deal with more general or different annotations; we shall add the relevant rules to treat public key cryptography in Section 8.

The LySA specification. We are now ready to systematically translate the annotated protocol narrations from Section 2 into LySA.

Protocol narrations usually focus on roles, i.e. initiator (A), responder (B), server (S). Actually, each principal may play many different roles. In the LySA specification we shall be explicit about using the protocol in a more general setting where many principals may use the protocol at the same time. We shall assume the existence of n+2 principals named I_i ($i \in \{\text{-}1, 0, 1, \cdots, n\}$); the name I_i can be thought of as the IP-address of the principal. Each of the principals $I_1, \cdots, I_n$ may serve in the initiator role of A as well as in the responder role of B; the principal I_i will present itself as (I_i, A) when serving as the initiator and as (I_i, B) when serving as the responder. We assume that there is *a single* server, which is modelled in the same way: its name is $I_{\text{-}1}$ and it will serve in the role of server S. Each principle can participate in an unlimited number of concurrent runs.

In the LySA specification we only explicitly describe the *legitimate* part of the system. Any principals *outside* the legitimate part (i.e. potential attackers) are given the canonical name I_0 and may take on any role.

The LySA specification of the WMF protocol is:

$$
\begin{array}{ll}
0. & (\nu^n_{i=1} K^A_i)(\nu^n_{j=1} K^B_j) \\
1. & |^n_{i=1} \; |^n_{\substack{j=1 \\ j \neq i}} \; !\,(\nu\, K_{ij}) \\
 & \quad \langle I_i, A, I_{-1}, S, I_i, A, \{I_j, B, K_{ij}\}^{A_i}_{K^A_i} [\mathsf{dest}\ S] \rangle . \\
3. & \quad (\nu\, m_{1ij}) \cdots (\nu\, m_{kij}) \\
 & \quad \langle I_i, A, I_j, B, \{m_{1ij}, \cdots, m_{kij}\}^{A_i}_{K_{ij}} [\mathsf{dest}\ B_j] \rangle \\
2'. & |\, |^n_{j=1} \,!\,(I_{-1}, S, I_j, B;\, y_j). \\
2''. & \quad |^n_{i=\text{-}1} \ \mathsf{decrypt}\ y_j\ \mathsf{as}\ \{I_i, A;\, y^K_{ij}\}^{B_j}_{K^B_j}\ [\mathsf{orig}\ S]\ \mathsf{in} \\
3'. & \quad\quad (I_i, A, I_j, B;\, z_{ij}). \\
3''. & \quad\quad \mathsf{decrypt}\ z_{ij}\ \mathsf{as}\ \{;\, z^{m_1}_{ij}, \cdots, z^{m_k}_{ij}\}^{B_j}_{y^K_{ij}}\ [\mathsf{orig}\ A_i]\ \mathsf{in}\ 0 \\
1'. & |\, |^n_{i=0} \,!\,(I_i, A, I_{-1}, S, I_i, A;\, x_i). \\
1''. & \quad |^n_{j=0}\ \mathsf{decrypt}\ x_i\ \mathsf{as}\ \{I_j, B;\, x^K_{ij}\}^S_{K^A_i}\ [\mathsf{orig}\ A_i]\ \mathsf{in} \\
2. & \quad\quad \langle I_{-1}, S, I_j, B, \{I_i, A, x^K_{ij}\}^S_{K^B_j} [\mathsf{dest}\ B_j] \rangle
\end{array}
$$

Here the checks of the extended narration above are performed by matching on inputs and decryptions.[2] It may be regarded as the main design goal of LySA, and of the translation of protocol narrations, that we separate the different branches of the matching to cater for a simple and efficient analysis.

The first line of the LySA specification ensures that the master keys between the legitimate principals and the server are unknown to outsiders; we assume that different keys K^A_i and K^B_i are used for the two roles of the principals. The same happens in line 1 for session keys, thereby taking care of the annotation [assuming K is a new key] put in the extended notation.

The next three lines model the principals I_i in their initiator roles. We let i range from 1 to n since we only model the legitimate part of the system. Each initiator wants to engage in a communication with any of the other *legitimate* principals I_j in their responder roles so we let j range from 1 to n as well.[3] An encrypted message sent from the principal I_i in the initiator role is labelled with the crypto-point A_i and annotated with the intended crypto-point for decryption such as S in line 1 above (strictly speaking the singleton set $\{S\}$); also, for the sake of readability we are overloading the symbol S to denote both the name of the server *and* a crypto-point, while we should instead use a different symbol for the second, e.g. ℓ_S. Correspondingly, the principal I_i in the responder role uses the crypto-point B_i, while the server uses the crypto-point S.

The next four lines model the legitimate principals I_j in their responder roles. The match of the first decryption (line $2''$) reveals the identity of the sender and here we are prepared to receive input from *any* agent i.e. we let j range from -1 to n. This follows the general scheme that whenever we do an input or a decryption we *never* restrict our attention to the legitimate part

[2] For simplicity, LySA only allows matching on *prefixes* of tuples. Consequently, we may occasionally have to rearrange the order of the elements of tuples though this has not been necessary for the WMF protocol.

[3] A principal does not want to authenticate itself; hence $j \neq i$. If needed, e.g. for checking confidentiality as in [44], we can remove this condition.

of the system; semantically our encoding is indistinguishable from writing one input, which matches the name of any principal.

The last three lines model the server. It is ready to handle requests from principles inside as well as outside the legitimate part of the system, and so $0 \leq i, j \leq n$; however, it does not accept messages from itself. Note that also agents outside the legitimate part of the system share master keys K_0^A and K_0^B with the server.

4 Control Flow Analysis

The aim of the analysis is to safely approximate when the reference monitor may abort the computation of a process P. The approximation is represented by a triple (ρ, κ, ψ) (resp. a pair (ρ, ϑ) when analysing a term E), called *estimate* for P (resp. for E), that satisfies the judgements defined by the axioms and rules of Table 2.

Terms. For each term E, the analysis will determine a *superset* of the possible canonical values that it may evaluate to. For this we keep track of the potential values of variables and to this end we introduce a global *abstract environment*:

- $\rho : \lfloor \mathcal{X} \rfloor \to \wp(\mathcal{V})$ maps the canonical variables to the sets of canonical values that they may be bound to.

Here we write $\mathcal{V}$ for the set of *canonical* terms with no free variables. The judgement for expressions takes the form

$$\rho \models E : \vartheta$$

and expresses that $\vartheta \subseteq \mathcal{V}$ is an acceptable estimate of the set of values that E may evaluate to in the abstract environment ρ. The judgement is defined by the axioms and rules in the upper part of Table 2. Note that we use the operation $\lfloor \cdot \rfloor$ to get hold of the canonical names and variables. Basically, the rules amount to demanding that ϑ contains all the canonical values associated with the components of a term; indeed, when $\mathsf{fv}(E) = \emptyset$ we have $\rho \models E : \{\lfloor E \rfloor\}$. In the sequel we shall use two kinds of membership tests: the usual $V \in \vartheta$ that simply tests whether V is in the set ϑ and the *faithful* test $V \,\mathsf{E}\, \vartheta$ that holds if there is a value V' in ϑ that equals V when the annotations are ignored, formally:

$$V \,\mathsf{E}\, \vartheta \quad \text{iff} \quad \exists V' \in \vartheta : [\![V]\!] = [\![V']\!]$$

Processes. In the analysis of processes we focus on which values can flow on the network:

- $\kappa \subseteq \wp(\mathcal{V}^*)$: the *abstract network environment* that includes all the message sequences that may flow on the network.

$$\frac{\lfloor n \rfloor \in \vartheta}{\rho \models n : \vartheta} \qquad \frac{\rho(\lfloor x \rfloor) \subseteq \vartheta}{\rho \models x : \vartheta}$$

$$\frac{\begin{array}{c}\wedge_{i=0}^{k}\, \rho \models E_i : \vartheta_i \wedge \\ \forall V_0, V_1, \cdots, V_k : \wedge_{i=0}^{k} V_i \in \vartheta_i \;\Rightarrow\; \{V_1, \cdots, V_k\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] \in \vartheta\end{array}}{\rho \models \{E_1, \cdots, E_k\}_{E_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] : \vartheta}$$

$$(\rho, \kappa) \models_{\mathsf{RM}} 0 : \psi \qquad \frac{(\rho, \kappa) \models_{\mathsf{RM}} P : \psi}{(\rho, \kappa) \models_{\mathsf{RM}} (\nu\, n)P : \psi}$$

$$\frac{\begin{array}{c}\wedge_{i=1}^{k}\, \rho \models E_i : \vartheta_i \wedge \\ \forall V_1, \cdots, V_k :\ \wedge_{i=1}^{k} V_i \in \vartheta_i \;\Rightarrow\; \langle V_1, \cdots, V_k \rangle \in \kappa \wedge \\ (\rho, \kappa) \models_{\mathsf{RM}} P : \psi\end{array}}{(\rho, \kappa) \models_{\mathsf{RM}} \langle E_1, \cdots, E_k \rangle.\, P : \psi}$$

$$\frac{\begin{array}{c}\wedge_{i=1}^{j}\, \rho \models E_i : \vartheta_i \wedge \\ \forall \langle V_1, \cdots, V_k \rangle \in \kappa :\ \wedge_{i=1}^{j} V_i \,\mathsf{E}\, \vartheta_i \Rightarrow\ \wedge_{i=j+1}^{k} V_i \in \rho(\lfloor x_i \rfloor) \wedge \\ (\rho, \kappa) \models_{\mathsf{RM}} P : \psi\end{array}}{(\rho, \kappa) \models_{\mathsf{RM}} (E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k).\, P : \psi}$$

$$\frac{(\rho, \kappa) \models_{\mathsf{RM}} P_1 : \psi \ \wedge\ (\rho, \kappa) \models_{\mathsf{RM}} P_2 : \psi}{(\rho, \kappa) \models_{\mathsf{RM}} P_1 | P_2 : \psi} \qquad \frac{(\rho, \kappa) \models_{\mathsf{RM}} P : \psi}{(\rho, \kappa) \models_{\mathsf{RM}} !\, P : \psi}$$

$$\frac{\begin{array}{c}\rho \models E : \vartheta \quad \wedge \quad \wedge_{i=0}^{j}\, \rho \models E_i : \vartheta_i \wedge \\ \forall\, \{V_1, \cdots, V_k\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] \in \vartheta :\ \wedge_{i=0}^{j} V_i \,\mathsf{E}\, \vartheta_i \ \Rightarrow \wedge_{i=j+1}^{k} V_i \in \rho(\lfloor x_i \rfloor) \wedge \\ (\neg \mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell, \ell') \in \psi) \wedge \\ (\rho, \kappa) \models_{\mathsf{RM}} P : \psi\end{array}}{(\rho, \kappa) \models_{\mathsf{RM}} \mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k\}_{E_0}^{\ell'}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P : \psi}$$

Table 2: Analysis of terms, $\rho \models E : \vartheta$, and analysis of processes, $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$.

To obtain this information we shall make use of the abstract environment ρ, as done for terms. The judgement for processes takes the form

$$(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$$

where ψ will be a possibly empty set of "error messages" of the form (ℓ, ℓ') indicating that something encrypted at ℓ was unexpectedly decrypted at ℓ'; we prove in Theorem 7 that when $\psi = \emptyset$ we may dispense with the reference monitor.

The judgement is defined by the axioms and rules in the lower part of Table 2 and is explained below.

Remember that the first three rules in Table 2 describe the analysis of terms and, thus, give the set of values ϑ that a term may evaluate to. This set is used e.g. in the rule for k-ary *output* that (i) finds the sets ϑ_i for each term E_i, (ii) requires that all k-tuples of values $\langle V_1, \cdots, V_k \rangle$ taken from $\vartheta_1 \times \cdots \times \vartheta_k$ can flow on the network, i.e that they are in the κ-component, and (iii) requires that (ρ, κ, ψ) are also valid analysis estimates of process P.

In the rule for *input* the terms $E_1, \cdots, E_j$ are used for matching values sent on the network. Thus, this rule (i) checks whether these first j terms have acceptable estimates ϑ_i and (ii) checks whether the first j values of any message $\langle V_1, \cdots, V_j, V_{j+1}, \ldots, V_k \rangle$ in κ (i.e. in any message predicted to flow on the network) are pointwise included in ϑ_i. The check is actually expressed using the faithful membership predicate, i.e. as $V_i \in \vartheta_i$, because annotations are ignored for matching just as in the semantics. If the check is successful then (iii) the values $V_{j+1}, \ldots, V_k$ are included in the estimates for the variables $x_{j+1}, \cdots, x_k$, respectively.

The rule for *decryption* handles the matching similarly to the rule for input: besides (i) establishing the validity of all the components of a decryption (i.e. the sets ϑ and ϑ_i) it (ii) checks for each encrypted value $\{V_1, \cdots, V_k\}^{\ell'}_{V_0}[\mathsf{dest}\ \mathcal{L}'] \in \vartheta$ whether the values $V_0, \ldots . V_j$ are pointwise included in the values in ϑ_i (including the key). Again we use the faithful membership tests for matching since the semantics ignores the annotations. If the check is successful then the values predicted for the variables x_i should pointwise contain the values V_i. Finally, (iii) the ψ-component of the analysis must contain (ℓ, ℓ') if the destination or origin assertions might be violated, i.e. if $(\ell \notin \mathcal{L}')$ or $(\ell' \notin \mathcal{L})$.

Both in the case of input and decryption we make sure only to analyse the continuation process P in those cases where the input or decryption could indeed succeed. This is essential for obtaining the necessary precision so that the analysis only rarely reports errors on correct protocols.

The rules for the inactive process, parallel composition, restriction and replication are straightforward.

Semantic properties. We prove below that our analysis respects the operational semantics of LySA. More precisely, we prove a subject reduction result for both the standard and the reference monitor semantics: if $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$, then the same triple (ρ, κ, ψ) is a valid estimate for all the states passed through

in a computation of P, i.e. for all the derivatives of P. Additionally, we show that when the ψ component is empty the reference monitor is useless; this is the basis to establish authentication.

It is convenient to prove the following lemmata. The first states that estimates are resistant to substitution of closed terms for variables, and it holds for both terms and processes. The second lemma says that an estimate for a process P is a valid estimate for every process congruent to P, as well.

Lemma 1 (Substitution)

- $\rho \models E : \vartheta$ *and* $\lfloor E' \rfloor \in \rho(\lfloor x \rfloor)$ *imply* $\rho \models E[E'/x] : \vartheta$.
- $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ *and* $\lfloor E' \rfloor \in \rho(\lfloor x \rfloor)$ *imply* $(\rho, \kappa) \models_{\mathsf{RM}} P[E'/x] : \psi$.

Lemma 2 (Congruence) *If* $P \equiv Q$ *then* $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ *iff* $(\rho, \kappa) \models_{\mathsf{RM}} Q : \psi$.

We are now ready to state the subject reduction result. It expresses that our analysis is semantically correct regardless of the way the semantics is parameterised.

Theorem 1 (Subject reduction)
If $P \rightarrow_{\mathcal{R}} Q$ *and* $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ *then also* $(\rho, \kappa) \models_{\mathsf{RM}} Q : \psi$.

The next result shows that our analysis correctly predicts when we can safely dispense with the reference monitor. We shall say that the reference monitor RM *cannot abort* a process P whenever there exist no Q, Q' such that $P \rightarrow^* Q \rightarrow Q'$ and $P \rightarrow^*_{\mathsf{RM}} Q \not\rightarrow_{\mathsf{RM}}$. As usual, $*$ stands for the transitive and reflexive closure of the relation in question, and $Q \not\rightarrow_{\mathsf{RM}}$ stands for $\neg\exists Q' : Q \rightarrow_{\mathsf{RM}} Q'$. We then have:

Theorem 2 (Static check for reference monitor)
If $(\rho, \kappa) \models_{\mathsf{RM}} P : \emptyset$ *then* RM *cannot abort* P.

Analysis of WMF. For the LySA specification of the WMF protocol given in Section 3 the minimal estimate (ρ, κ, ψ) satisfying

$$(\rho, \kappa) \models_{\mathsf{RM}} \mathsf{WMF} : \psi$$

is given by $\psi = \emptyset$ and will have the following non-empty entries (for $1 \leq i, j \leq n$, $i \neq j$, and $1 \leq l \leq k$) for ρ

$$\begin{array}{lrcl}
\rho : & x_i & \mapsto & \{\{I_j, B, K_{ij}\}^{A_i}_{K_i^A}[\mathsf{dest}\ S]\} \\
& x_{ij}^K & \mapsto & \{K_{ij}\} \\
& y_j & \mapsto & \{\{I_j, A, K_{ij}\}^{S}_{K_i^B}[\mathsf{dest}\ B_j]\} \\
& y_{ij}^K & \mapsto & \{K_{ij}\} \\
& z_{ij} & \mapsto & \{\{m_{1ij}, \cdots, m_{kij}\}^{A_i}_{K_{ij}}[\mathsf{dest}\ B_j]\} \\
& z_{ij}^{m_l} & \mapsto & \{m_{lij}\}
\end{array}$$

whereas κ is

$$\begin{array}{ll}\kappa: & \{\langle I_i, A, I_{-1}, S, I_i, A, \{I_j, B, K_{ij}\}^{A_i}_{K_i^A}[\mathsf{dest}\ S]\rangle\}\\ & \cup\{\langle I_{-1}, S, I_j, B, \{I_i, A, K_{ij}\}^{S}_{K_j^B}[\mathsf{dest}\ B_j]\rangle\}\\ & \cup\{\langle I_i, A, I_j, B, \{m_{1ij}, \cdots, m_{kij}\}^{A_i}_{K_{ij}}[\mathsf{dest}\ B_j]\rangle\}\end{array}$$

Observe that y^K_{ij} is bound to the session key K_{ij} and that $z^{m_l}_{ij}$ is bound to m_{lij} indicating the communication of m_{lij} from principal I_i to principal I_j.

5 Modelling the Attacker

Protocols are executed in an environment where there may be malicious attackers. Writing P_{sys} for the implementation of the protocol, the actual environment may take the form of an arbitrary process having a placeholder for P_{sys} as further elaborated in [17]; for most process algebras the *characteristic contexts* take the form $P_{sys} \mid Q$ for some process Q representing the environment and this is the scenario we consider as well.

Hardest attackers and Dolev-Yao. We aim at finding a formula $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ characterising all attackers; this means that whenever an estimate (ρ, κ, ψ) satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ then $(\rho, \kappa) \models_{\mathsf{RM}} Q : \psi$ for all attackers Q. There are at least two approaches to finding such a formula. One is to define a formula inspired by the pioneering studies of Dolev and Yao [34] and then to prove its correctness. The other is to find a "hardest attacker" and to prove that it is as strong as any other attacker as was done for firewall security in [55]. We base our presentation on the first approach and then show the close connection between the two approaches in Theorem 9.

To characterise all attackers we need to make a few assumptions that benefit the control flow analysis but that have no semantic consequences. We shall say that a process P is of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ whenever: (1) it is closed (i.e. has no free variables), (2) its *free* names are in $\mathcal{N}_{\mathsf{f}}$, (3) all the arities used for sending or receiving are in $\mathcal{A}_\kappa$ and (4) all the arities used for encryption or decryption are in $\mathcal{A}_{\mathsf{Enc}}$. Clearly we can inspect P_{sys} to find minimal $\mathcal{N}_{\mathsf{f}}$, $\mathcal{A}_\kappa$, $\mathcal{A}_{\mathsf{Enc}}$ such that P_{sys} is of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ and then P_{sys} is also of type $(\mathcal{N}'_{\mathsf{f}}, \mathcal{A}'_\kappa, \mathcal{A}'_{\mathsf{Enc}})$ provided that $\mathcal{N}_{\mathsf{f}} \subseteq \mathcal{N}'_{\mathsf{f}}$, $\mathcal{A}_\kappa \subseteq \mathcal{A}'_\kappa$ and $\mathcal{A}_{\mathsf{Enc}} \subseteq \mathcal{A}'_{\mathsf{Enc}}$. To avoid having to deal with too many special cases we shall assume that $\mathcal{A}_\kappa$ and $\mathcal{A}_{\mathsf{Enc}}$ contain the number 1.

Given $\mathcal{A}_{\mathsf{Enc}}$ we write k_{Enc} for the minimal positive integer that dominates all elements of $\mathcal{A}_{\mathsf{Enc}}$, i.e. $k_{\mathsf{Enc}} = \min\{k > 0 \mid \forall k' \in \mathcal{A}_{\mathsf{Enc}} : k' \leq k\}$, and then we write $\mathcal{A}^+_{\mathsf{Enc}} = \mathcal{A}_{\mathsf{Enc}} \cup \{k_{\mathsf{Enc}} + 1\}$ for $\mathcal{A}_{\mathsf{Enc}}$ enlarged with the arity $k_{\mathsf{Enc}} + 1$ of permitted encryptions. We claim that when studying a system P_{sys} of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ there is no loss of generality in assuming that attackers Q have type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$; in particular we claim that the ability of the attacker to use:

- additional free names may be masked by restricting the names so as to become local within Q,

(1) $\wedge_{k\in\mathcal{A}_\kappa}\ \forall\langle V_1,\cdots,V_k\rangle\in\kappa : \wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)$
(2) $\wedge_{k\in\mathcal{A}^+_{\mathsf{Enc}}}\ \forall\{V_1,\cdots,V_k\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}]\in\rho(z_\bullet) :$
$V_0\ \mathsf{E}\ \rho(z_\bullet)\Rightarrow(\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\wedge(\neg\mathsf{RM}(\ell,\mathcal{C},\ell_\bullet,\mathcal{L})\Rightarrow(\ell,\ell_\bullet)\in\psi))$
(3) $\wedge_{k\in\mathcal{A}^+_{\mathsf{Enc}}}\ \forall V_0,\cdots,V_k : \wedge_{i=0}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow\{V_1,\cdots,V_k\}^{\ell_\bullet}_{V_0}[\mathsf{dest}\ \mathcal{C}]\in\rho(z_\bullet)$
(4) $\wedge_{k\in\mathcal{A}_\kappa}\ \forall V_1,\cdots,V_k : \wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow\langle V_1,\cdots,V_k\rangle\in\kappa$
(5) $\{n_\bullet\}\cup\lfloor\mathcal{N}_{\mathsf{f}}\rfloor\subseteq\rho(z_\bullet)$

Table 3: Dolev-Yao condition.

- a "private channel" based on k-ary communication for $k\notin\mathcal{A}_\kappa$ does not increase its computational power, and
- a "private channel" based on k-ary encryptions and decryptions for $k\notin\mathcal{A}_{\mathsf{Enc}}$ may be coded using the ability to perform nested $k_{\mathsf{Enc}}+1$-ary encryptions and decryptions.

The actual definition of $k_{\mathsf{Enc}}+1$ is not so important; what is important is that it is a number k such that $k>1$ and $k\notin\mathcal{A}_{\mathsf{Enc}}$.

One difficulty concerning attackers is that we have no control over the canonical names and variables used. This motivates inspecting P_{sys} to find the finite set $\mathcal{N}_{\mathsf{c}}$ of all canonical names used and the finite set $\mathcal{X}_{\mathsf{c}}$ of all canonical variables used. We then postulate a new canonical name $n_\bullet$ not in $\mathcal{N}_{\mathsf{c}}$ and a new canonical variable $z_\bullet$ not in $\mathcal{X}_{\mathsf{c}}$. Given a process Q of type $(\mathcal{N}_{\mathsf{f}},\mathcal{A}_\kappa,\mathcal{A}^+_{\mathsf{Enc}})$ we then construct the semantically equivalent process $\overline{Q}'$ as follows: (a) all restrictions $(\nu\, n)P$ are α-converted (in the classical sense) into restrictions $(\nu\, n')P'$ where n' has the canonical representative $n_\bullet$, (b) all occurrences of variables x_i in inputs $(E_1,\cdots,E_j;\, x_{j+1},\cdots,x_k).\,P$ and in decryptions
$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1,\cdots,E_j;\, x_{j+1},\cdots,x_k\}^{\ell}_{E_0}\ [\mathsf{orig}\ \mathcal{L}]\ \mathsf{in}\ P$ are α-converted (in the classical sense) to use variables x'_i with canonical representative $z_\bullet$. Thus, $\overline{Q}'$ only uses finitely many *canonical* names and variables.

Another aspect concerning attackers is the presence of annotations. In our view the attacker really should not have annotations at encryption and decryption points since the annotations are intended for expressing the intentions of the protocol and the attacker cannot be part of this. However, the syntax forces us to place annotations everywhere and we therefore take the semantically equivalent approach of ensuring that all annotations are the trivial ones, $[\mathsf{dest}\ \mathcal{C}]$ and $[\mathsf{orig}\ \mathcal{C}]$, and that all crypto-points are replaced by the crypto-point $\ell_\bullet$ not occurring in P_{sys}. We write $\overline{\overline{Q}}$ for the resulting process.

We now have sufficient control over the capabilities of the attacker that we can characterise the potential effect of all attackers $\overline{\overline{Q}}$ of type $(\mathcal{N}_{\mathsf{f}},\mathcal{A}_\kappa,\mathcal{A}^+_{\mathsf{Enc}})$. We do so by defining the formula $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ of type $(\mathcal{N}_{\mathsf{f}},\mathcal{A}_\kappa,\mathcal{A}^+_{\mathsf{Enc}})$ for expressing the Dolev-Yao condition for LySA; it is defined as the conjunction of the five components in Table 3 (actually, three more components are added later on in Table 9 to cope with public key encryption).

The formula in Table 3 makes it clear that the attacker initially has some knowledge (5), that it may learn more by eavesdropping (1) or by decrypting messages with keys already known (2), that it may construct new encryptions using the keys known (3) and that it may actively forge new communications (4).

We can now establish the correctness of the Dolev-Yao condition for LySA. We first show that the estimates satisfying $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ are valid for all attackers, thus proving a sort of "soundness" of the Dolev-Yao condition.

Theorem 3 (Soundness of Dolev-Yao condition)
If (ρ, κ, ψ) satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ then $(\rho, \kappa) \models_{\mathsf{RM}} \overline{Q} : \psi$ for all attackers Q of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$.

We can now show the close connection between the Dolev-Yao condition and the notion of "hardest attackers" [55]. This result also shows the "completeness" of the Dolev-Yao condition: we have not needlessly added capabilities that cannot be possessed by real attackers.

Theorem 4 (Existence of "Hardest Attacker")
There exists an attacker Q_{hard} of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ such that the formula $(\rho, \kappa) \models_{\mathsf{RM}} \overline{Q_{hard}} : \psi$ is equivalent to the formula $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$.

Crypto-based authenticity. The annotations of LySA were designed to facilitate studying the properties of *origin authentication* and of *destination authentication*. The first property amounts to making sure that an encrypted message that principal B_j expects from principal A_i (that we write on the form $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}^\ell_{E_0}\ [\mathsf{orig}\ A_i]\ \mathsf{in}\ P$) was indeed encrypted *only* by A_i. The second property is symmetric: a message that A_i intends for B_j (written $\{E_1, \cdots, E_k\}^\ell_{E_0}[\mathsf{dest}\ B_j]$) is successfully decrypted *only* by B_j.

More formally, for the dynamic property we say that P_{sys} guarantees *dynamic authentication* with respect to the annotations in P_{sys} if the reference monitor RM cannot abort $P_{sys} \mid \overline{Q}$ regardless of the choice of the attacker Q.

Similarly, for the static property we say that P_{sys} guarantees *static authentication* with respect to the annotations in P_{sys} if there exists ρ and κ such that $(\rho, \kappa) \models_{\mathsf{RM}} P_{sys} : \emptyset$ and $(\rho, \kappa, \emptyset)$ satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$. As will become clear in Section 6, the actual test that we have implemented considers the minimal type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ of P_{sys} and computes the minimal solution (ρ, κ, ψ) such that$(\rho, \kappa) \models_{\mathsf{RM}} P_{sys} : \psi$ and (ρ, κ, ψ) satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ and finally tests whether or not $\psi = \emptyset$.

Theorem 5 (Authentication) *If P_{sys} guarantees static authentication then P_{sys} guarantees dynamic authentication.*

Attack 1	Attack 2
$A \to S: \quad A, \{B, K\}_{K_A}$ $S \to M_B: A, \{K\}_{K_B}$ $M_S \to B: A', \{K\}_{K_B}$ $A \to B: \quad \{m_1 \cdots m_k\}_K$	$M \to S: \quad M, \{B, K\}_{K_M}$ $S \to M_B: M, \{K\}_{K_B}$ $M_S \to B: A', \{K\}_{K_B}$ $M \to B: \quad \{m_1 \cdots m_k\}_K$

Attack 3	Attack 4	Attack 5
$A \to M_S: \quad A, B, \{K\}_{K_A}$ $M_A \to S: \quad A, B', \{K\}_{K_A}$ $S \to B': \quad \{A, K\}_{K_{B'}}$ $A \to M_B: \quad \{m_1 \cdots m_k\}_K$ $M_A \to B': \{m_1 \cdots m_k\}_K$	$A \to M_S: A, B, \{K\}_{K_A}$ $M_S \to S: A, M, \{K\}_{K_A}$ $S \to M: \quad \{A, K\}_{K_M}$ $A \to M_B: \{m_1 \cdots m_k\}_K$	$A \to M_S: A, B, \{K\}_{K_A}$ $M_S \to S: A, M, \{K\}_{K_A}$ $S \to M: \quad \{A, K\}_{K_M}$ $M_A \to S: A, B, \{K\}_{K_A}$ $S \to B: \quad \{A, K\}_{K_B}$ $M \to B: \quad \{m_1 \cdots m_k\}_K$

Table 4: Attacks on WMF variations.

Validation of WMF. We analyse the WMF protocol of Section 2 and restrict our attention to the solutions that satisfy the formula $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ thereby taking care of the Dolev-Yao attacker. The least solution has an empty ψ-component reflecting that the analysis guarantees static as well as dynamic authentication.

We now consider two variants of the protocol: one where the initiator's name is not encrypted and one where the responder's name is not encrypted (see Appendix A). In the first case the ψ-component is

$$\{(A_i, B_j) \mid i \neq j, 1 \leq i, j \leq n\} \ \cup \ \{(\ell_\bullet, B_j) \mid 1 \leq j \leq n\}$$

showing that static authentication fails. The pair (A_i, B_j) shows that a value encrypted at A_i has wrongfully been decrypted at B_j; similarly, the pair $(\ell_\bullet, B_j)$ shows that a value created by the attacker has been decrypted at B_j. Actually also dynamic authentication fails: the two contributions to ψ correspond to the first two attack sequences of Table 4; here we write M_X to denote the attacker (called M) pretending to be X. Both sequences will result in B believing that he is communicating with A' although he is communicating with A and M, respectively.

In the case where the responder's name is not encrypted (see Appendix A) the ψ component becomes

$$\begin{array}{l}\{(A_i, B_j) \mid 1 \leq i, j \leq n\} \cup \\ \{(A_i, \ell_\bullet) \mid 1 \leq i \leq n\} \ \cup \ \{(\ell_\bullet, B_j) \mid 1 \leq j \leq n\}\end{array}$$

so again the analysis shows that static authentication fails. Also dynamic authentication fails: the attacks corresponding to the three contributions to ψ are shown in the last three columns of Table 4.

6 The Implementation

One can show that for any given P there always is a least choice of ρ, κ, and ψ such that $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ and (ρ, κ, ψ) satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ (and accordingly a least

choice of each ϑ whenever $\rho \models E : \vartheta$ is required by the analysis of P). The aim of our implementation is to compute such a least (ρ, κ, ψ) for any given LySA process P. However, the analysis components ρ, κ, and ϑ are interpreted over the *infinite* universe of terms and this poses the main challenge in obtaining a terminating implementation. As detailed below, we handle this challenge by encoding sets of terms as generating tree grammars; essentially representing an infinite set by a *finite* number of grammar rules.

Our implementation follows the standard strategy for implementing constraint based program analysis [54]: it encodes the analysis into a suitable constraint language and use a standard constraint solver to compute the least solution to these constraints. To obtain an efficient implementation we use the Succinct Solver [57] as our constraint solver. It computes the least interpretation of predicate symbols within "constraints" written in Alternation-free Least Fixed Point logic (ALFP). This is an expressive fragment of first order predicate logic that is interpreted over a finite universe, which fits well with our encodings of infinite sets as finite tree grammars. The encoding of the analysis into ALFP proceeds in a number of steps as follows.

Firstly, the specification of the analysis in Table 2 is *succinct* [59]; this means that ψ and ϑ are local in the judgements in which they occur. Using the techniques of [58] the analysis is transformed into a *verbose* specification [59]; this means that ψ and ϑ become global components. This is obtained by adding further labels to the syntax and making every analysis component global by using the new labels to link the values of the components to specific places in the syntax.

Secondly, by applying techniques from [56], sets of terms are conceived as languages generated by tree grammars. A first step is to transform the analysis so that ρ and κ both contain *labels* of terms rather that the terms themselves and leave ϑ to be the only component that contains sets of terms. The second step is to represent the sets in ϑ as (regular, normalised) tree grammars [27] over signatures where k-ary encryptions are represented by $k+1$-ary constructors and the canonical names are represented as constants (i.e. 0-ary constructors). The tree grammars will use the *labels* as non-terminals and have rules where the left-hand side is a label while the right-hand side is a constructor applied to labels. As an example consider the LySA process (where labels are written as superscripts and ignoring for the moment the annotations of crypto-points):

$$P \quad = \quad \langle n^{l_1} \rangle.\,\mathsf{0} \mid !\,(;\, x).\,\langle \{x^{l_2}\}^{l_4}_{k^{l_3}} \rangle.\,\mathsf{0}$$

The process sends the terms $n, \{n\}_k, \{\{n\}_k\}_k, \{\{\{n\}_k\}_k\}_k, \ldots$ over the network and in doing so it binds the variable x to each of these values. The analysis with sets of terms are encoded as tree grammars in ϑ will then specify that

$$\begin{array}{llll} \kappa & \supseteq & \{l_1, l_4\} & \qquad \vartheta: \; l_1 \to \lfloor n \rfloor \qquad l_3 \to \lfloor k \rfloor \\ \rho(x) & \supseteq & \{l_1, l_4\} & \qquad\quad\; l_2 \to \lfloor n \rfloor \qquad l_4 \to \{l_2\}_{l_3} \\ & & & \qquad\quad\; l_2 \to \{l_2\}_{l_3} \end{array}$$

That is, all the terms that can be generated from label l_1 and label l_4 may be sent on the network as described by κ and may also be bound to x as described by ρ.

The term that may be generated from these labels can be found by inspecting the grammar rules in ϑ. For example, the language generated by starting at l_1 is $\{\lfloor n \rfloor\}$ while the grammar for the language generated from l_2 contains a circularity producing the infinite set $\{\lfloor n \rfloor, \{\lfloor n \rfloor\}_{\lfloor k \rfloor}, \{\{\lfloor n \rfloor\}_{\lfloor k \rfloor}\}_{\lfloor k \rfloor}, \ldots\}$. These are of course precisely the values that x may be bound to during the execution of P.

Thirdly, the encoding proceeds by transforming the analysis into ALFP. This involves a number of straightforward encodings such as representing sets as predicates and encoding the finite sequences used in communication and encryption into predicates of a fixed arity.

Finally, the analysis is turned into a generation function, $\mathcal{G}$, such that $\mathcal{G}(P)$ is an ALFP formula that represents the analysis of P. In addition to satisfying the $\mathcal{G}(P)$, the analysis estimates also need to satisfy $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ for the attacker. To obtain this, we take advantage of the attacker process Q_{hard} (which is a "hardest attacker" as described in the proof of Theorem 9 in Appendix C) and the *conjunction* of $\mathcal{G}(P)$ *and* $\mathcal{G}(Q_{hard})$ is passed on to the Succinct Solver. In turn, this calculates an encoding of the estimate (ρ, κ, ψ) for P *in combination with* the attacker. A more detailed description of the implementation together with proofs of soundness may be found in [16]. Here soundness means that the least solution to the encoding of the analysis also provides a (not necessarily least) solution to the original analysis.

The time complexity of solving a formula in the Succinct Solver is polynomial in the size of the universe, over which the formula is interpreted. For our implementation the universe is linear in the size of the process and a simple worst-case estimate of the degree of the complexity polynomial is given as one plus the maximal nesting depth of quantifiers in the formula [57]. For our current implementation the nesting depth is governed by the maximal length of the sequences used in communication and encryption though techniques from [56] might have been be applied to yield a cubic worst-case upper bound; we have refrained from doing so because in practice the implementation runs in sub-cubic time.

7 Validation of Protocols

In this section we summarise the analysis results we have obtained for a number of variations of the following symmetric key protocols: Wide Mouthed Frog (as studied in Section 2) [5, 18], Needham-Schroeder [52], Amended Needham-Schroeder [53], Otway-Rees [60], Yahalom [18] and Andrew Secure RPC [67]. The protocol narrations are summarised in Appendix A. In the actual experiments we have taken the number of principals, n, to be 3.

Robustness of protocol narrations. In our formalisation of protocol narrations in LySA in Section 2 we decided to focus on: *(i)* separating identities (e.g. I_i) from roles (e.g. A and B), and *(ii)* using distinct master keys (e.g. K_i^A and K_i^B) for distinct roles. Decisions like these are crucial for the properties

protocol $1 \leq i, j \leq n,\ i \neq j$	$A \neq B$ $\wedge_{i=0}^{n} K_i^A \neq K_i^B$	$A = B$ $\wedge_{i=0}^{n} K_i^A \neq K_i^B$	$A \neq B$ $\wedge_{i=0}^{n} K_i^A = K_i^B$	$A = B$ $\wedge_{i=0}^{n} K_i^A = K_i^B$
Wide Mouthed Frog	$\emptyset$	$\emptyset$	$\emptyset$	$(A_i, B_i), (S, S)$
with nonces	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Needham-Schroeder	(A_i, A_i)	(A_i, A_i)	(A_i, A_i)	(A_i, A_i)
with type flaw corrected	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Amended Needham-Schroeder	(A_i, A_i)	(A_i, A_i)	(A_i, A_i)	(A_i, A_i)
with type flaw corrected	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Otway-Rees	$\emptyset$	$\emptyset$	$(B_i, S), (S, B_i)$	$(B_i, S), (S, B_i)$
Yahalom	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
with BAN optimisation	$\emptyset$	$\emptyset$	$\emptyset$	(A_i, B_i), (S, A_i), (S, B_i)
Paulson's amendment	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Andrew Secure RPC	(A_i^3, B_j^1), (B_i^2, A_j^4)	(A_i^3, B_j^1), (B_i^2, A_j^4)	(A_i^3, B_j^1), (B_i^2, A_j^4)	(A_i^3, B_j^1), (B_i^2, A_j^4)
with BAN correction and type flaw corrected	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Table 5: Overview of results: robustness of protocol narrations.

of the protocol and our first experiment will show not only that some of the protocols are more robust than others but also that our approach is able to pinpoint the amount of safeguarding needed for a protocol to be trustworthy (see Table 5).

As shown in the first column of Table 5, when roles as well as master keys are kept distinct we observe a non-empty value for ψ in the case of Needham-Schroeder. This reflects a potential problem due to a type flaw so that the attacker sends the incremented nonce (in step 5 of Appendix A) instead of the nonce (in step 4). A simple correction is to insert unique tags in the encrypted messages produced in the protocol; with these corrections our analysis result reports that the problem has disappeared. Similar type flaws and corrections are observed for Amended Needham-Schroeder and Andrew Secure RPC.

The next three columns of Table 5 show what happens when we omit some of the safeguards. For the Wide Mouthed Frog, ψ is non-empty when master keys and roles are the same. This corresponds to a *parallel session* attack (reported in e.g. [35]) where the first message from one session gets mixed up with the second message from another. The attack cannot take place when we keep roles or master keys apart, which is confirmed by the analysis result.

For Otway-Rees there is an attack when master keys are not kept distinct. This corresponds to an attack reported in [61], which exploits that encrypted messages from a principal acting both as initiator and responder may be confused.

In [18] an optimised version of the Yahalom protocol is suggested and an attack is reported in [70]; from Table 5 we see that the attack only succeeds when not distinguishing between roles and when using the same master key for distinct roles. Paulson [62] suggests an amendment whose correctness we can

protocol	K_{12}^{old} is leaked
Wide Mouthed Frog	$(\ell_\bullet, B_2)$
with nonces	$\emptyset$
Needham-Schroeder	$(B_2, \ell_\bullet), (\ell_\bullet, B_2)$
with type flaw corrected	
Amended Needham-Schroeder	$\emptyset$
with type flaw corrected	
Otway-Rees	$\emptyset$
Yahalom	$(\ell_\bullet, B_2)$
with BAN optimisation	$\emptyset$
Paulson's amendment	$\emptyset$
Andrew Secure RPC	$(A_1, \ell_\bullet)$
with type flaw corrected	
with BAN correction	$\emptyset$

Table 6: Overview of results: leaking of old session keys.

confirm.

Our findings suggest that many classical problems such as parallel session, type flaw, and reflection attacks occur precisely because a number of crucial distinctions are not made sufficiently clear in the protocol narrations; it is encouraging to observe that our approach can pinpoint this.

Leaking an old session key. Many protocols become insecure when old session keys are compromised. Our second experiment shows that our approach is able to detect also these vulnerabilities. To be specific we add an old session key K_{12}^{old} and the corresponding tickets issued by the server to the knowledge of the attacker in formula (5) in the definition of the formula $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$ in Section 5. We perform our experiments using full safeguards: roles and identities are kept distinct and distinct master keys are used for distinct roles (see Table 6).

Our results confirm that the WMF protocol as presented in [5] is problematic when old session keys are leaked. The original presentation in [18] used time stamps, but since we do not model time, we present a correction with nonces (see Appendix A); our analysis result then guarantees static as well as dynamic authentication even in the presence of leaked old session keys.

We also confirm that the Needham-Schroeder protocol is vulnerable to the leaking of old session keys; the corresponding attack is that of Denning-Sacco [32]. Furthermore, our analysis results guarantee static and dynamic authentication for the Amended Needham-Schroeder protocol (with the type flaw corrected) and the Otway-Rees protocol in the presence of leaked old session keys.

For the Yahalom protocol our analysis result shows that there may be an authentication problem in case of leaked old session keys. This is a false alarm which is due to the independent attribute nature of our analysis. It is interesting to observe that, although the authenticity of the protocol has been proved by

Paulson in [62], he mentions that the proof is considerable more complex than that for the BAN optimised version and that he had to introduce a relation keeping track of associated pairs of session keys and responder nonces; in our terminology this would correspond to introducing a relational component in the analysis [54]. For the BAN optimised version (and the amendment suggested by Paulson) our analysis guarantees static as well as dynamic authentication in the presence of leaked old session keys.

For the Andrew Secure RPC protocol we observe the problem with leaks of old keys as reported in [18]; our analysis confirms that the amended version suggested in [18] indeed solves the problem.

8 Asymmetric Cryptography

Our framework can be easily adapted to deal with perfect asymmetric cryptography [33]. In this scheme, keys are no longer shared between principals. Each party P is associated instead with some key pair (m^+, m^-). Usually, the key m^+ is called *public* and it is made available to everybody, while the *private* key m^- is assumed to be known to P, only. Any principal willing to send some ciphered message to P can encrypt it by using m^+; only P can then decrypt it using m^-. Some public key algorithms, such as RSA [65], allows for a complementary use of the key pair: the owner of m^- *digitally signs* a message by encrypting it with his private key, and the receiver checks the signature by decrypting the message with the corresponding public key m^+. Below, we only give the needed extensions to LySA and its analysis in order to treat this kind of cryptography; a treatment of digital signatures more in agreement with current practice may be found in [17].

8.1 Extensions to the Calculus

The syntax is enriched as follows. First, we add to the set of names enough pairs of asymmetric keys m^+, m^-:

$$N ::= n \mid m^+ \mid m^-$$

Terms now include these new names and a construct for asymmetric encryptions, quite similar to that for symmetric encryption, that carries the same decorations. The only difference is that the key may turn out to be public or private, so accounting for both encryption and signature.

$E ::=$	*terms*
$\cdots$	$\cdots$
m^+, m^-	public and private keys
$\{\!\lvert E_1, \cdots, E_k \rvert\!\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]$	asymmetric encryption ($k \geq 0$)

The syntax of processes P has the following two additional constructs, one for creating a new pair of public/private keys and the other for asymmetric

decryption or signature validation, with the usual labels and annotations.

$P ::=$	*processes*
$\ldots$	$\ldots$
$(\nu_{\pm}\, m)P$	key pair creation
$\mathsf{decrypt}\ E\ \mathsf{as}\ \{\!\lfloor E_1, \cdots, E_j; x_{j+1}, \cdots, x_k \rfloor\!\}^{\ell}_{E_0}\ [\mathsf{orig}\ \mathcal{L}]\ \mathsf{in}\ P$	asymmetric decryption ($k \geq 0$)

Notational conventions and assumptions on canonical values are the same. In particular, $\lfloor m \rfloor$,$\lfloor m^+ \rfloor$,$\lfloor m^- \rfloor$ are pairwise distinct. Note that $\mathsf{fn}(N) = N$.

The rules for substitution are extended by the following:

- $[n/m]N = \begin{cases} n & \text{if } N = m \\ N & \text{otherwise} \end{cases}$
- $[n^+, n^-/m^+, m^-]N = \begin{cases} n^+ & \text{if } N = m^+ \\ n^- & \text{if } N = m^- \\ N & \text{otherwise} \end{cases}$

We need some further congruence rules and two new clauses for defining the semantics of the just introduced processes. The additional rules for congruence are:

- $(\nu_{\pm}\, m)\mathsf{0} \equiv \mathsf{0}$;
- $(\nu_{\pm}\, n)(\nu_{\pm}\, m)P \equiv (\nu_{\pm}\, m)(\nu_{\pm}\, n)P$;
- $(\nu\, n)(\nu_{\pm}\, m)P \equiv (\nu_{\pm}\, m)(\nu\, n)P$;
- $(\nu_{\pm}\, m)(P \mid Q) \equiv P \mid (\nu_{\pm}\, m)Q$, if $m^+, m^- \notin \mathsf{fn}(P)$.

The new reduction rules are in Table 7. The rule (A-Res) is quite standard. As it was the case for symmetric decryption in rule (Decr), the process $\mathsf{decrypt}\, E$ $\mathsf{as}\{\!\lfloor E'_1, \cdots, E'_j; x_{j+1}, \cdots, x_k \rfloor\!\}^{\ell'}_{E'_0}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\, P$ attempts to decrypt E, provided that $E = \{\!\lfloor E_1, \cdots, E_k \rfloor\!\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]$ such that (E_0, E'_0) is a pair consisting of a public key and its private counterpart (it is irrelevant which is which, so catering for both asymmetric encryption and for digital signature validation *à la* RSA[65]), and that $\llbracket E_i \rrbracket = \llbracket E'_i \rrbracket$ for all $1 \leq i \leq j$. If the conditions hold then the process behaves as $P[E_{j+1}/x_{j+1}, \ldots, E_k/x_k]$.

8.2 Extensions to the Control Flow Analysis

The additional rules for the control flow analysis are in Table 8 and are very similar to the corresponding rules (2), (3) and (5) in Table 2. The only differences occur in the rule for *asymmetric decryption*: the values V_0, V'_0 are actually a pair of public/private keys, required by the condition $\{V_0, V'_0\} = \{m^+, m^-\}$, and the consequent check for $1 \leq i \leq j$, that the values V_i are pointwise included in the values in ϑ_i.

$$\text{(A-Res)}\quad \frac{P \rightarrow_{\mathcal{R}} P'}{(\nu_{\pm}\, m)P \rightarrow_{\mathcal{R}} (\nu_{\pm}\, m)P'}$$

$$\text{(A-Decr)}\quad \frac{\wedge_{i=1}^{j} [\![E_i]\!] = [\![E_i']\!] \quad \wedge \quad \{E_0, E_0'\} = \{m^+, m^-\} \quad \wedge \quad \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{\begin{array}{l}\mathsf{decrypt}\ \{\!|E_1, \cdots, E_k|\!\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]\ \mathsf{as}\ \{\!|E_1', \cdots, E_j'; x_{j+1}, \cdots, x_k|\!\}^{\ell'}_{E_0'}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P \\ \quad \rightarrow_{\mathcal{R}} P[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]\end{array}}$$

Table 7: Semantic rules for asymmetric cryptography.

$$\frac{\lfloor m^+ \rfloor \in \vartheta}{\rho \models m^+ : \vartheta} \qquad \frac{\lfloor m^- \rfloor \in \vartheta}{\rho \models m^- : \vartheta}$$

$$\frac{\begin{array}{c}\wedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \\ \forall V_0, V_1, \cdots, V_k : \wedge_{i=0}^{k} V_i \in \vartheta_i \ \Rightarrow\ \{\!|V_1, \cdots, V_k|\!\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta\end{array}}{\rho \models \{\!|E_1, \cdots, E_k|\!\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}] : \vartheta}$$

$$\frac{(\rho, \kappa) \models_{\mathsf{RM}} P : \psi}{(\rho, \kappa) \models_{\mathsf{RM}} (\nu_{\pm}\, m)P : \psi}$$

$$\frac{\begin{array}{c}\rho \models E : \vartheta \quad \wedge \quad \wedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \\ \forall (m^+, m^-) : \forall\, \{\!|V_1, \cdots, V_k|\!\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta : \forall V_0' \in \vartheta_0 : \{V_0, V_0'\} = \{m^+, m^-\} \wedge \\ \wedge_{i=1}^{j} V_i \in \vartheta_i \ \Rightarrow\ \wedge_{i=j+1}^{k} V_i \in \rho(\lfloor x_i \rfloor) \wedge \\ \qquad (\neg \mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell, \ell') \in \psi) \wedge \\ \qquad (\rho, \kappa) \models_{\mathsf{RM}} P : \psi\end{array}}{(\rho, \kappa) \models_{\mathsf{RM}} \mathsf{decrypt}\ E\ \mathsf{as}\ \{\!|E_1, \cdots, E_j; x_{j+1}, \cdots, x_k|\!\}^{\ell'}_{E_0}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P : \psi}$$

Table 8: Analysis of new constructs for asymmetric cryptography.

(6) $\forall (m^+, m^-) : \wedge_{k \in \mathcal{A}^+_{\mathsf{Enc}}} \forall \{\!|V_1, \cdots, V_k|\!\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \rho(z_\bullet) :$
$\forall V_0' \in \rho(z_\bullet) : \{V_0, V_0'\} = \{m^+, m^-\} \Rightarrow (\wedge_{i=1}^{k} V_i \in \rho(z_\bullet) \wedge$
$(\neg \mathsf{RM}(\ell, \mathcal{C}, \ell_\bullet, \mathcal{L}) \Rightarrow (\ell, \ell_\bullet) \in \psi))$

(7) $\wedge_{k \in \mathcal{A}^+_{\mathsf{Enc}}} \forall V_0, V_1, \cdots, V_k : \wedge_{i=0}^{k} V_i \in \rho(z_\bullet) \Rightarrow \{\!|V_1, \cdots, V_k|\!\}^{\ell_\bullet}_{V_0}[\mathsf{dest}\ \mathcal{C}] \in \rho(z_\bullet)$

(8) $\{m_\bullet^+, m_\bullet^-\} \subseteq \rho(z_\bullet)$

Table 9: Additional Dolev-Yao conditions for asymmetric cryptography.

Finally, we extend the Dolev-Yao conditions for the asymmetric case, as shown by Table 9. Again, there are very little differences with the symmetric case; note that we postulate a new pair of canonical names $m_\bullet^+, m_\bullet^-$, and that the rule (5) in Table 3 already considers the symmetric keys and the special canonical name $n_\bullet$.

All the results given in Section 4 and 5 hold also for the asymmetric case. More in detail, both the substitution and the congruence lemmata, both the theorems on subject reduction and on the static check for the reference monitor, and both the theorems on soundness of the Dolev-Yao condition and on the existence of a "hardest attacker" hold. Their statements and proofs in this general case are given in Appendix C.

The annotations added for asymmetric key cryptography are rather similar to those used for symmetric key cryptography although there are differences. This is because keys are intended to be kept secret in a symmetric key setting while one of the keys in a key pair is intended to be *public* in the asymmetric setting. Hence we need to reconsider the methodology described in Section 2 for which set of labels to use when annotating protocol narrations and LySa processes.

When encrypting a message with a public key, an annotation [dest ℓ] is added to the encrypted term to describe the unique place where the message is intended to be decrypted. This is completely analogous to the symmetric case. When decrypting a message that has been encrypted using a public key, a corresponding annotation, such as [orig $\ell, \cdots$], should be added. Clearly we should include the label indicating the place where the encryption is intended to have been made, but it may be necessary also to add labels for other places (including those within the attacker) where the public key may have been used for encryption. Since LySa combines decryption and pattern matching, it is often the case that a specific decryption (and the embedded pattern matching) should only succeed for a subset of messages that might have been encrypted with the public key. In many well-behaved protocols the intention is that it is not possible to confuse any two messages for one another, i.e. that the origination point in the protocol should be unique. This argument hardly extends to the completely uncontrolled behaviour of the attacker. Hence for well-behaved protocols is is often safe to use annotations of the form [orig $\{\ell, \ell_\bullet\}$] which state that the encryption is intended to have been performed at a unique point within the protocol (as indicated by ℓ) or within the attacker (as indicated by $\ell_\bullet$). This is indeed the scheme we have used for our experiments.

In the case of signature validation, the situation is reversed, since messages encrypted with a private key may be decrypted by anyone knowing the public key and, in particular, by the attacker. As far as the attacker is concerned we continue our strategy of assuming that there are no annotations in the attacker; this is already reflected in Table 9.

Our experimental validation of protocols using asymmetric key cryptography is reported in Table 10. First, we consider two versions of the classical Needham Schroeder Public Key protocol [52] as given by the narrations in Appendix A. The first version is the full 7 messages protocol as given in [52] that relies on

a server for delegation of public keys while the other is a 3 message version [45] that has no key server and instead uses the scope of $(\nu_{\pm}\, m)P$ to model key delegation. As shown in Table 10 both versions of the protocol exhibit a flaw where A_i wrongfully accepts a message encrypted by some B_j though A_i was actually trying to authenticate with the attacker. This message is part of the attack reported by Lowe [45] and we should note that the attack only occurs if we allow A_i to initiate a session with the attacker as in Lowe's attack. In both versions of the protocol the amendment suggested by Lowe in [45] results in an empty ψ, thereby, validating the correctness of the amendment. Since both versions behave in a similar manner we furthermore conclude that the delegation of public keys may equally well be modelled by the scope of $(\nu_{\pm}\, m)P$ as by the use of a key server.

Wireless information networks and portable communication technology offer us another interesting scenario to test our technique. We analyse the protocol by Beller, Chang, and Yacobi [8], called MSR (for Modular Square Root). In this setting, communication between portable devices A's and ports (or base stations) B's is wireless and each port serves more than one portable at the same time. A certification authority U is requested to sign certificates, using its private key K_U^-. In the protocol, the portable A wants to be authenticated by the base station B. More precisely, on receiving from B its public key K_B, the portable A uses it to encrypt the new session key K and send it to B. Afterwards, to authenticate the new session key to B, A sends also its identity and a secret certificate encrypted (symmetric cryptography) with K. The secret certificate, $\{\!|A|\!\}_{K_U^-}$ must be signed by a certification authority U.

The encoding of this protocol slightly diverges from our standard encoding presented in Section 3, since portables and base stations are clearly distinct entities and cannot play the roles of one another. Hence, the protocol is encoded in a framework where the portables have identities I_i while the base stations have distinct identities J_j.

The MSR protocol is only meant to authenticate portables to base stations and not vice versa. As pointed out by Carlsen [22] this is *not* achieved by the protocol because the attacker may masquerade as a base station. This attack turns up as the pairs $(A_i, \ell_\bullet)$ and $(\ell_\bullet, B_j)$ in ψ in Table 10. Additionally, we found an undocumented family of attacks on this protocol. Essentially, the attacker forces a portable to wrongly authenticate with an arbitrary port within the reach of the attacker, as reported by the pair (A_i, B_j). Strictly speaking, this attack does not compromise the one-way authentication intended by the MSR protocol. The Improved MSR (IMSR) protocol [8], on the other hand, is meant to provide mutual authentication by including an extra certificate for the base station B in the first message ($\{\!|B|\!\}_{K_U^-}$). Once the certificate has been sent in clear it may, however, be replayed as reported by Carlsen [22]. As a consequence all the attacks on MSR still work on the IMSR and this includes our new parallel session attack where unintended use of the new certificate shows up as pairs (B_j, A_i) in ψ.

To be explicit we may construct the parallel session attack as follows. First

protocol	ψ for $1 < i, j < n$
Needham-Schroeder Public Key	(B_j, A_i)
Lowe's amendment	$\emptyset$
Needham-Schroeder Public Key – no server	(B_j, A_i)
Lowe's amendment	$\emptyset$
Modular Square Root	(A_i, B_j), $(A_i, \ell_\bullet)$, $(\ell_\bullet, B_j)$
Improved Modular Square Root	(A_i, B_j), $(A_i, \ell_\bullet)$, $(\ell_\bullet, B_j)$, (B_j, A_i)

Table 10: Experiments with protocols using asymmetric cryptography.

B_1 and B_2 both start sessions with A:

$$\begin{array}{lll}
2.1 & B_2 \to M_A: & B_2, K^+_{B_2} \\
1.1 & B_1 \to M_A: & B_1, K^+_{B_1} \\
1.1' & M_{B_1} \to A: & B_1, K^+_{B_2} \\
1.2 & A \to M_{B_1}: & \{\!|K_{11}|\!\}_{K^+_{B_2}} \\
1.3 & A \to M_{B_1}: & \{A, \{\!|A|\!\}_{K^-_U}\}_{K_{11}} \\
2.2' & M_A \to B_2: & \{\!|K_{11}|\!\}_{K^+_{B_2}} \\
2.3' & M_A \to B_2: & \{A, \{\!|A|\!\}_{K^-_U}\}_{K_{11}}
\end{array}$$

At the end A thinks it is talking to B_1 but really A is talking to B_2.

9 Related Work

Many papers have considered process algebras such as CSP, CCS, π- and Spi-calculus as a suitable medium for expressing protocols in a more precise manner than the usual protocol narrations yet in a considerably more abstract and succinct manner than in real programming languages. A number of strong techniques based on logical theories have successfully been developed for confidentiality (e.g. formalisations of Dolev and Yao [14, 34, 61]) and authenticity [18] issues; many have been investigated with the help of proof assistants or automatic theorem provers, although the computational overhead is rather high, and with the help of constraint solvers. Also the use of type (and effect) systems have become popular as a technique for analysing such systems — as a companion to logical methods based on inference systems or semantic methods based on testing equivalence or bisimulations. Static analysis based on control flow analysis has shown promise of analysing such systems, and here we have been able to demonstrate its ability to find flaws in protocols. Further, non exhaustive details of these developments are provided below.

The modal logic of beliefs BAN [18] plays a central role in the logical approach to protocol analysis. The discovery of many known and unknown protocol flaws witnesses its great success and many extensions have been proposed in

the literature (see e.g. [40, 6, 71]). Indeed, BAN logic is particularly expressive, although it essentially focuses on authentication; additionally, it is very easy to write protocol specifications and to reason about them. However, these specifications are not mechanised and sometimes the resulting formulae loose the operational flavour of protocols. This may make it hard to establish soundness with an operational semantics — completeness does not generally hold. Particularly relevant is also the role of initial beliefs: unrealistic or wrong assumptions may lead to unexpected results.

Quite successful in analysing known protocols is another approach based on general purpose logics and (time consuming) theorem provers. Bolignano [14] models the intended behaviour of the honest principals separately from the Dolev-Yao attacker, specified as a set of inductive formulae. Authentication properties refer to suitable temporal relations between the events (i.e. occurrences of the steps) of the protocol. Also Paulson [61] inductively defines the attacker, while the formalisation of a protocol consists of the set of all possible traces of events. The properties are proved by induction on traces, and according to Paulson, the proofs may happen to be quite long. Still in the logical approach, there is the systematic derivation of security protocols proposed in [29] based on the process algebra CORDS [36], equipped with a logic for reasoning about properties of specifications. We also mention the method for rapid prototyping and analysis of protocols developed in Maude [30].

More oriented to an operational approach, there is the model checking approach; see e.g. Millen's *Interrogator* [49], Mitchell's *murϕ* [51] and Clarke's Brutus [25]. The method relies on the construction of the model, a finite state automaton that represents the behaviour of the protocol. In most cases the automata are sequential, but also causality is exploited, see e.g. the Athena checker [69], based on strand spaces [72]. The proofs then consist of an exhaustive search of the automaton and of the verification that each reachable state enjoys the desired property, expressed as a formula in a modal temporal logic. The model checkers are very fast, but to have a finite, small search space, some simplifying assumptions are needed.

The CAPSL project [48, 31] is based on a detailed high level language for expressing protocols, seen as a list of messages sent (but without any prescription for their recipients). Besides this, a user declares the objects involved, and writes assumptions and assertions about their intended use, e.g. secrecy, authentication, freshness, and may also specify the operating environment. The resulting programs are given a semantics by a translation into a Horn fragment of linear logic. The proofs of security properties, essentially trace invariants, are then carried out on this intermediate representation, exploiting various tools (e.g. inductive verifiers, model checkers, an efficient PROLOG based constraint solver [50], further improved by [28]). Set constrains with equality are also used in [26] to decide (in double exponential time) secrecy properties.

The AVISS project [7] provides a protocol verification framework that is very similar to CAPSL though their back-end validation tools rely on ideas from finite state model checking, only. These techniques may efficiently find flaws in protocols but, in contrast to our approach, they have no general guarantee of

termination in case the protocol is correct. To ensure termination they adopt a notion of bounded validation of protocols (saying that no flaws have been found within a certain bounded subset of all executions).

The NRL protocol analyser by Meadows [47] is perhaps the first automatic tool for verifying security properties. It combines the features of theorem provers with those of constraint solvers, and it has been used to successfully analyse several protocols. Recently the analyser has been extended to cope with explicit cryptographic primitives, through suitable unification algorithms, and then used in [23] to verify the Internet Key Exchange protocol that involves the Diffie-Hellman key exchange schema.

Lowe [45] specifies protocols in CSP and exploits its operational semantics to construct their (finite) models that are then checked using FDR [66]. This approach lead to the discovery of the man-in-the-middle attack in the public key Needham-Schroeder protocol that was so far considered secure. In the same vein, see also [68].

After the seminal paper by Lowe, many different process algebras have been used to formalise and verify protocols in an operational setting. Gorrieri and Focardi [38] used CCS and formalised the non-interference security property as a bisimulation property. Also they verified a large number of protocols automatically. In a subsequent paper [39], authentication *à la* Woo-Lam [73] has been shown to be a special case of non-interference.

Abadi and Gordon proposed the Spi-calculus [5] and used a testing equivalence to guarantee security properties. Checking this equivalence is problematic for infinite-state processes, and some symbolic techniques have been proposed in these cases [15, 37]. Abadi used type systems for secrecy in [1]; since then, many papers follow this research line, using both the Spi- [3, 4], the distributed π- [63, 64], and the ambient-calculus [20, 21, 19]. More recently, Gordon and Jeffrey [41, 42, 43] defined type (and effects) systems that statically guarantee authentication of protocols specified in a Spi-calculus enriched with assertions *à la* Woo-Lam. They report on a number of flawed and correct protocols checked by their technique. Blanchet [9] also uses such assertions on a similar calculus and provides a tool that may be seen as an implementation of type inference cf. [4]. His tool has, in general, no guarantee of termination, which is the major difference from our polynomial-time implementation. Each of the above proposals address a single property at a time, and any new property requires a different type system. Although the following point is seldom discussed, these systems seem to lack principal types, so only efficient type checkers seem to be available, while type inference is computationally intractable.

In [12] we proposed a control flow analysis for the π-calculus; we then extended it to the Spi-calculus in [13]. The analysis results have been used to show that some protocols obey some security policies, including confidentiality and variations of Bell and LaPadula's Mandatory Access Control [12] and of non-interference [10]. The format of our analysis is robust: only little additions are needed, e.g., to pass from the π- to the Spi-calculus and to deal with terms. The flexibility of our technique has been demonstrated on the ambient calculus in [55]. These papers focus on establishing the validity of firewalls, and introduce

the notion of "hardest attacker" used in Section 5.

The main points we see in favour of the static analysis approach taken here are the following: (*i*) that solutions do always exist and are computed in low polynomial time; (*ii*) that the analysis is correct w.r.t. a formal operational semantics (in the form of a subject reduction theorem, just as for type systems); and (*iii*) that a single analysis suffices for a variety of properties: different inspections of a solution permit to check different security properties of a protocol, with no need of re-analysing it several times.

10 Conclusion

We have shown that protocol narrations may be formalised as LySA-processes such that a static analysis can pinpoint a wide variety of errors in communication protocols, both documented and undocumented.

The calculus. The design of LySA was patterned after the Spi-calculus but LySA has been adapted so as to facilitate that useful information may be obtained from a relatively unsophisticated static analysis. Extensions of the analysis may be able to deal directly with a more permissive calculus.

We have taken a perfect view of cryptography, both symmetric and asymmetric. We only considered attacks or phenomena that can be expressed in LySA. Since *time* is not present in LySA we cannot deal with the duration of time stamps, i.e. when they do not merely serve as nonces. Because we only allow *structured data* we do not deal with bit strings and type flaw attacks based on a concatenation of two bit strings being viewed as a single bit string; in our view a diligent use of Abstract Syntax Notation One (ASN.1) will provide the necessary safe guards. Following the development of [17], we can deal with a more direct treatment of hash functions, message authentication codes, and digital signatures and certificates.

The security properties. We have focused on authentication properties based on *origin authentication* and *destination authentication*. This notion of authentication has the advantage that it can directly be captured by the operational semantics and therefore also by a static analysis. In our view we capture many of the authenticity problems normally studied using a session-based approach, i.e. where certain end-of-transactions need to match with the right begin-of-transactions. Including the specification of security goals in our narrations, is somewhat reminiscent of the Woo and Lam's ideas of correspondence assertions [73] or the annotations added in the idealised protocol narrations in BAN logic [18].

Our techniques are also able to deal with a number of other security properties, e.g. secrecy can be checked in the manner of [12]. In the present development this amounts to inspect the contents of $\rho(z_\bullet)$ in order to determine whether or not "secrets" may end up in the attacker. Partitioning values in secret and public then suffices: if $\rho(z_\bullet)$ only contains public values confidentiality

is guaranteed. Alternatively one may extend LySA with explicit confidentiality annotations: whenever a new name is introduced we add the set $\mathcal{X}' \subseteq \mathcal{X}$ of canonical variables to which the name may be bound, e.g. $(\nu\, N[\mathsf{within}\mathcal{X}'])$, and at each binding occurrence we add the set of canonical names $\mathcal{N}' \subseteq \mathcal{N}$ that may be bound to the variable, e.g. $(\cdots; x_i[\mathsf{from}\ \mathcal{N}'], \cdots)$.

Moving further in the direction of annotations we may add beliefs in the style of BAN logic. For example, we may decide to change the syntax of LySA to add annotations to the creation of new nonces about its intended use, e.g. $(\nu\, N[A \rightarrow B])$ might denote the creation of a nonce intended to establish an authentic connection from A to B. The main challenge will be to modify the reference monitor; there may well be BAN-like annotations where it is unclear how to enforce them by means of a reference monitor.

The static analysis. We have made an effort in choosing a static analysis that is informative, that has good performance and that is not overly complicated to explain. We would like to extend the analysis to deal with the *multiplicities* of messages in order deal with replay attacks also from the same round and with more general correspondence properties than the non-injective agreement considered here. Also we would like to add additional information to the analysis that would facilitate constructing the *finite counterexamples* that constitute the real proof of the existence of protocol flaws (as in Table 6). Finally, we would like to extend the analysis to be transition-oriented in order to deal more directly with session-based authentication properties in the manner of Woo and Lam [73].

Acknowledgements. We wish to thank Colin Boyd for kind discussion on MSR flaws.

References

[1] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 5(46):18–36, 1999.

[2] M. Abadi. Security protocols and specifications. In *Proc. of FoSSaCS'99*, LNCS 1578, pages 1–13. Springer, 1999.

[3] M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. In *Proc. of FoSSaCS'01*, LNCS 2030, pages 25–41. Springer, 2001.

[4] M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. In *Proc. of POPL'02*, pages 33–44. ACM Press, 2002.

[5] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols - The Spi calculus. *Information and Computation 148*, 1:1–70, 1999.

[6] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, 1991.

[7] A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS security protocol analysis tool. In *CAV'02*, LNCS 2404, pages 349–353. Springer, 2002.

[8] M. J. Beller, L.-F. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. *IEEE Journal of Selected Areas in Communications*, 11(6):821–829, 1993.

[9] B. Blanchet. From secrecy to authenticity in security protocols. In *Proc. of SAS'02*, LNCS 2477, pages 342–359. Springer, 2002.

[10] C. Bodei. *Security Issues in Process Calculi.* PhD thesis, Dipartimento di Informatica, Università di Pisa. TD-2/00, March, 2000.

[11] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. Riis Nielson. Automatic validation of protocol narration. In *Proc. of CSFW'03*, pages 126–140. IEEE, 2003.

[12] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for the π-calculus with applications to security. *Information and Computation*, 168:68–92, 2001.

[13] C. Bodei, P. Degano, H. Riis Nielson, and F. Nielson. Flow logic for Dolev-Yao secrecy in cryptographic processes. *Future Generation Computer Systems*, 18(6):747–756, 2002.

[14] D. Bolignano. An approach to the formal verification of cryptographic protocols. In *Proc. of 3rd ACM Conf. on Computer and Communications Security*, pages 106–118. ACM Press, 1996.

[15] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. of ICALP'01*, LNCS 2305, pages 667–681. Springer, 2001.

[16] M. Buchholtz. Implementing control flow analysis for security protocols. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, To appear. Preliminary version available at http://www.imm.dtu.dk/∼mib/lysa/toALFP.pdf.

[17] M. Buchholtz, F. Nielson, and H. Riis Nielson. A calculus for control flow analysis of security protocols. *International Journal of Information Security*, To appear.

[18] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, pages 18–36, 1990.

[19] L. Cardelli, G. Ghelli, and A. D. Gordon. Ambient groups and mobility types. In *Proc. of International IFIP Conference TCS 2000*, LNCS 1872, pages 333–347. Springer, 2000.

[20] L. Cardelli and A.D. Gordon. Mobile ambients. In *Proc. of FoSSaCS'98*, LNCS 1378, pages 140–155. Springer, 1998.

[21] L. Cardelli and A.D. Gordon. Types for mobile ambients. In *Proc. of POPL'99*, pages 79–92. ACM Press, 1999.

[22] U. Carlsen. Optimal Privacy and Authentication on a Portable Communications System. *Operating Systems Review*, 28(3):16–23, 1994.

[23] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. Analysis of the internet key exchange protocol using the NRL protocol analyser. In *Proc. of Symposium on Security and Privacy*, pages 216–231. IEEE, 1999.

[24] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proc. of CSFW'99*. IEEE, 1999.

[25] E.M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with Brutus. *ACM Trans. on S/W Engineering and Methodology*, 9(4):443–487, 2000.

[26] H. Comon, V. Cortier, and J. Mitchell. Tree automata with memory, set constraints and ping pong protocols. In *Proc. of ICALP'01*, LNCS 2305. Springer., 2001.

[27] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. http://www.grappa.univ-lille3.fr/tata/, 27th September 2002.

[28] R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In *Proc. of SAS'02*, LNCS 2477. Springer., 2002.

[29] A. Datta, A. Derek, J. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proc. of CSFW'03*, pages 109–125. IEEE, 2003.

[30] G. Denker, J. Meseguer, and C. Talcott. Protocol specification and analysis in Maude. In *Proc. of Workshop on Formal Methods and Security Protocols*, 1998.

[31] G. Denker and J. Millen. CAPSL integrated protocol environment. In *DARPA Information Survivability Conference (DISCEX 2000)*, pages 207–221. IEEE, 2000.

[32] D. E. Denning and G. M. Sacco. Timestamps in key distribution systems. *CACM*, 24(8):533–536, 1981.

[33] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[34] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, 1983.

[35] A. Durante, R. Focardi, and R. Gorrieri. A compiler for analysing cryptographic protocols using non-interference. *ACM Transactions on Software Engineering and Methodology*, 9(4):488–528, 2000.

[36] N. Durgin, J. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *Proc. of CSFW'01*, pages 241–255. IEEE, 2001.

[37] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc. of CSFW'01*, pages 160–173. IEEE, 2001.

[38] R. Focardi and R. Gorrieri. A classification of security properties. *Journal of Computer Security*, 3(1), 1995.

[39] R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Proc. of ICALP'00*, LNCS 1853. Springer, 2000.

[40] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proc. of Symposium on Research in Security and Privacy*, pages 234–248. IEEE, 1990.

[41] A. D. Gordon and A. Jeffrey. Authenticity by Typing for Security Protocols. In *Proc. of CSFW'01*. IEEE, 2001.

[42] A. D. Gordon and A. Jeffrey. Typing Correspondence Assertions for Communication Protocols. In *Proc. of Mathematical Foundations of Programming Semantics*, 2001.

[43] A. D. Gordon and A. Jeffrey. Types and Effects for Asymmetric Cryptographic Protocols. In *Proc. of CSFW'02*, pages 77 –91, 2002.

[44] J. Heather and S. Schneider. Towards automatic verification of authentication protocols on an unbounded network. In *Proc. of CSFW'00*, pages 132 – 143. IEEE, 2000.

[45] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[46] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. of CSFW '97*, pages 18–30. IEEE, 1997.

[47] C. Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[48] J. Millen. CAPSL web site. http://www.csl.sri.com/users/millen/capsl/.

[49] J. Millen. The Interrogator: A tool for cryptographic protocol security. In *Proc. of Symposium on Security and Privacy*, pages 134–141. IEEE, 1984.

[50] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communication Security*, pages 166–175. ACM SIGSAC, 2001.

[51] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murϕ. In *Proc. of Conference on Security and Privacy*, pages 141–153. IEEE, 1997.

[52] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[53] R. M. Needham and M. D. Schroeder. Authentication revisited. *ACM Operating Systems Review*, 21(1):7–7, 1987.

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

[55] F. Nielson, H. Riis Nielson, and R. R. Hansen. Validating firewalls using flow logics. *Theoretical Computer Science*, 283(2):381–418, 2002.

[56] F. Nielson, H. Riis Nielson, and H. Seidl. Cryptographic analysis in cubic time. *Electronic Notes of Theoretical Computer Science*, 62, 2002.

[57] F. Nielson, H. Riis Nielson, and H. Seidl. A succinct solver for ALFP. *Nordic Journal of Computing*, 9:335–372, 2002.

[58] H. Riis Nielson and F. Nielson. Flow logics for constraint based analysis. In *Proc. of CC'98*, LNCS 1383, pages 109–127. Springer, 1998.

[59] H. Riis Nielson and F. Nielson. Flow Logic: a multi-paradigmatic approach to static analysis. In *The Essence of Computation: Complexity, Analysis, Transformation*, LNCS 2566, pages 223–244. Springer, 2002.

[60] D. Otway and O. Rees. Efficient and timely mutual authentication. *ACM Operating Systems Review*, 21(1):8–10, 1987.

[61] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[62] L. C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. Technical report, Cambridge University, England, 1998.

[63] J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proc. of POPL'98*, pages 378–390. ACM Press, 1998.

[64] J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. In *Proc. of POPL'99*, pages 93–104. ACM Press, 1999.

[65] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosytems. *Communications of the ACM*, 21(2):120–126, 1978.

[66] B. Roscoe and P. Gardiner. Security Modelling in CSP and FDR: Final Report. Technical report, Formal Systems Europe, 1995.

[67] M. Satyanarayanan. Integrating security in a large distributed system. *ACM ToCS*, 7(3):247–280, 1989.

[68] S. Schneider. Security properties and CSP. In *Proc. of Symposium on Research in Security and Privacy*, pages 174–187. IEEE, 1996.

[69] D.X. Song. Athena: a new efficient automatic checker for security protocol analysis. In *Proc. of CSFW'99*, pages 192–202. IEEE, 1999.

[70] P. Syverson. A taxonomy of replay attacks. In *Proc. of CSFW'94*, pages 187–191. IEEE, 1994.

[71] P. Syverson and P. van Oorschot. A unified cryptographic protocol logic. Technical report, NRL Publication 5540-227. Naval Research Lab, 1996.

[72] F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. of Conference on Security and Privacy*, pages 160–171, Los Alamitos, 1998. IEEE.

[73] T.Y.C. Woo and S.S. Lam. A semantic model for authentication protocols. In *Proc. of Symposium on Security and Privacy*, pages 178–194. IEEE, 1993.

A Protocol Narrations

The analysis results reported in the paper are based on the following versions of the protocols. The corresponding LySA specifications are obtained following the guidelines of Section 2.

Wide Mouthed Frog. [5]

1. $A \to S: \quad A, \{B, K\}_{K_A}$
2. $S \to B: \quad \{A, K\}_{K_B}$
3. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

The initiator's name is not encrypted:

2. $S \to B: \quad A, \{K\}_{K_B}$

The responder's name is not encrypted:

1. $A \to S: \quad A, B, \{K\}_{K_A}$

The pair operation is modelled by an encryption with the key PAIR as explained in Section 3.

A version with nonces:

1. $A \to B: \quad A, N_A$
2. $B \to S: \quad \{A, B, (N_A, K)\}_{K_B}$
3. $S \to A: \quad \{B, (N_A, K)\}_{K_A}$
4. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

Needham-Schroeder (symmetric key). [52]

1. $A \to S: \quad A, B, N_A$
2. $S \to A: \quad \{N_A, B, K, \{K, A\}_{K_B}\}_{K_A}$
3. $A \to B: \quad \{K, A\}_{K_B}$
4. $B \to A: \quad \{N_B\}_K$
5. $A \to B: \quad \{N_B{+}1\}_K$
6. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

Correcting the type flaw:

4. $B \to A: \quad \{u_1, N_B\}_K$
5. $A \to B: \quad \{u_2, N_B{+}1\}_K$

The successor operation is modelled by an encryption with the key SUCC that is known to the attacker. The type flaw is corrected by inserting extra components $(u_1, u_2, \cdots)$ in the encrypted messages.

Amended Needham-Schroeder. [53]

1. $A \to B: \quad A$
2. $B \to A: \quad \{A, N'_B\}_{K_B}$
3. $A \to S: \quad A, B, N_A, \{A, N'_B\}_{K_B}$
4. $S \to A: \quad \{N_A, B, K, \{K, N'_B, A\}_{K_B}\}_{K_A}$
5. $A \to B: \quad \{K, N'_B, A\}_{K_B}$
6. $B \to A: \quad \{N_B\}_K$
7. $A \to B: \quad \{N_B{+}1\}_K$
8. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

Correcting the type flaw:

6. $B \to A: \quad \{u_1, N_B\}_K$
7. $A \to B: \quad \{u_2, N_B{+}1\}_K$

Otway-Rees. [60]

1. $A \to B: \quad N, \{N_A, N, A, B\}_{K_A}$
2. $B \to S: \quad N, \{N_A, N, A, B\}_{K_A}, \{N_B, N, A, B\}_{K_B}$
3. $S \to B: \quad N, \{N_A, K\}_{K_A}, \{N_B, K\}_{K_B}$
4. $B \to A: \quad N, \{N_A, K\}_{K_A}$
5. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

Yahalom. [18]

1. $A \to B: \quad A, N_A$
2. $B \to S: \quad B, \{A, N_A, N_B\}_{K_B}$
3. $S \to A: \quad \{B, K, N_A, N_B\}_{K_A}, \{A, K\}_{K_B}$
4. $A \to B: \quad \{A, K\}_{K_B}, \{N_B\}_K$
5. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

BAN optimised version:

1. $A \to B: \quad A, N_A$
2. $B \to S: \quad B, N_B, \{A, N_A\}_{K_B}$
3. $S \to A: \quad N_B, \{B, K, N_A\}_{K_A}, \{A, K, N_B\}_{K_B}$
4. $A \to B: \quad \{A, K, N_B\}_{K_B}, \{N_B\}_K$
5. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

Paulson's amendment [62]:

3. $S \to A: \quad N_B, \{B, K, N_A\}_{K_A}, \{A, B, K, N_B\}_{K_B}$
4. $A \to B: \quad \{A, B, K, N_B\}_{K_B}, \{N_B\}_K$

Andrew Secure RPC. [67]

1. $A \to B: \quad A, \{N_A\}_K$
2. $B \to A: \quad \{N_A{+}1, N_B\}_K$
3. $A \to B: \quad \{N_B{+}1\}_K$
4. $B \to A: \quad \{K', N'_B\}_K$
5. $A \to B: \quad \{m_1, \cdots, m_k\}_{K'}$

Correcting the type flaw:

1. $A \to B: \quad A, \{u_1, N_A\}_K$
2. $B \to A: \quad \{u_2, N_A{+}1, N_B\}_K$
3. $A \to B: \quad \{u_3, N_B{+}1\}_K$
4. $B \to A: \quad \{u_4, K', N'_B\}_K$

BAN corrections:

4. $B \to A: \quad \{u_4, K', N'_B, N_A\}_K$

For this protocol we use unique crypto-points in the LySA specification in order to get a more precise account of the errors reported by the analysis.

Needham-Schroeder Public Key. [52]

1. $A \to S: \quad A, B$
2. $S \to A: \quad \{\!|B, K_B^+|\!\}_{K_S^-}$
3. $A \to B: \quad \{\!|A, N_A|\!\}_{K_B^+}$
4. $B \to S: \quad B, A$
5. $S \to B: \quad \{\!|A, K_A^+|\!\}_{K_S^-}$
6. $B \to A: \quad \{\!|N_A, N_B|\!\}_{K_A^+}$
7. $A \to B: \quad \{\!|N_B|\!\}_{K_B^+}$

Lowe's amendment [45]:

6. $B \to A: \quad \{\!|B, N_A, N_B|\!\}_{K_A^+}$

No server:

3. $A \to B: \quad \{\!|A, N_A|\!\}_{K_B^+}$
6. $B \to A: \quad \{\!|N_A, N_B|\!\}_{K_A^+}$
7. $A \to B: \quad \{\!|N_B|\!\}_{K_B^+}$

Lowe's amendment [45]:

6. $B \to A: \quad \{\!|B, N_A, N_B|\!\}_{K_A^+}$

MSR [8]

1. $B \to A: \quad B, K_B^+$
2. $A \to B: \quad \{\!|K|\!\}_{K_B^+}$
3. $A \to B: \quad \{A, \{\!|A|\!\}_{K_U^-}\}_K$

Improved MSR [8]

1. $B \to A: \quad B, \{\!|B|\!\}_{K_U^-}, K_B^+$
2. $A \to B: \quad \{\!|K|\!\}_{K_B^+}$
3. $A \to B: \quad \{A, \{\!|A|\!\}_{K_U^-}\}_K$

B Comparison to the Spi-calculus

Ignoring the annotations, the encryption construct of the Spi-calculus [5] corresponds to that of LySA and the standard decryption operation $\mathsf{decrypt}\ E\ \mathsf{as}\ \{x_1, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$ corresponds to $\mathsf{decrypt}\ E\ \mathsf{as}\ \{;\ x_1, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$. In the other

direction our notion of decryption can be modelled in the Spi-calculus by a decryption followed by a number of explicit matchings. In more detail, the LySA process

$$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1,\cdots,E_j;\ x_{j+1},\cdots,x_k\}_{E_0}\ \mathsf{in}\ P$$

behaves as the Spi-process:

$$\begin{array}{l}\mathsf{decrypt}\ E\ \mathsf{as}\ \{x_1,\cdots,x_j,x_{j+1},\cdots,x_k\}_{E_0}\\ \mathsf{in}\ [x_1=E_1]\cdots[x_j=E_j]P.\end{array}$$

We need to extend LySA in order to precisely code input and output of the Spi-calculus. Indeed, coding the monadic output $\overline{c}\langle v\rangle.P$ of the Spi-calculus as $\langle c,v\rangle.\,P$ and the corresponding input $c(x).P$ as $(c;\,x).\,P$ only works if syntactic means are used to ensure that *all* processes match at least one component in each input.

LySA may further be extended with a concept of channels as discussed in [17]. Finally, in LySA we do not seem to have a need for the matching construct $[E=E']P$ of the Spi-calculus; in an extended calculus it can be coded as $(\nu c)(\overline{c}\langle E\rangle.\,\mathsf{0}\mid c(E';\,).\,P)$.

C Proofs

In this appendix restates the lemmata and theorems presented earlier in the paper and gives the proofs of their correctness. The statements in this appendix should be interpreted over the full calculus including the constructs for asymmetric cryptography in Section 8 and, thus, the proofs are given for this general case. Note, however, that the statements are formulated in precisely the same way as for the restricted calculus in Section 4 and Section 5 and of course also hold in the more restricted setting.

Lemma 3 (Substitution)

- $\rho\models E:\vartheta$ *and* $\lfloor E'\rfloor\in\rho(\lfloor x\rfloor)$ *imply* $\rho\models E[E'/x]:\vartheta$.
- $(\rho,\kappa)\models_{\mathsf{RM}} P:\psi$ *and* $\lfloor E'\rfloor\in\rho(\lfloor x\rfloor)$ *imply* $(\rho,\kappa)\models_{\mathsf{RM}} P[E'/x]:\psi$.

Proof. Both proofs are by a straightforward structural induction. □

Lemma 4 (Congruence) *If* $P\equiv Q$ *then* $(\rho,\kappa)\models_{\mathsf{RM}} P:\psi$ *iff* $(\rho,\kappa)\models_{\mathsf{RM}} Q:\psi$.

Proof. It suffices to inspect the rules for $\equiv$ and to recall that the analysis only involves canonical names; that it ignores restrictions; that associativity and commutativity of $\wedge$ reflects the same properties of $\mid$ and that any triple is a valid estimate for $\mathsf{0}$. □

Theorem 6 (Subject reduction)
If $P\rightarrow_{\mathcal{R}} Q$ *and* $(\rho,\kappa)\models_{\mathsf{RM}} P:\psi$ *then also* $(\rho,\kappa)\models_{\mathsf{RM}} Q:\psi$.

Proof. We prove the more general result

- $(\rho,\kappa) \models_{\mathsf{RM}} P : \psi$ and $P \rightarrow_{\mathcal{R}} Q$ imply $(\rho,\kappa) \models_{\mathsf{RM}} Q : \psi$; furthermore, if $\psi = \emptyset$ then $P \rightarrow_{\mathsf{RM}} Q$.

by induction on the inference $P \rightarrow_{\mathcal{R}} Q$ (as given in Table 1 and Table 7).

In case (Com) we assume
$(\rho,\kappa) \models \langle E_1,\cdots,E_k\rangle.\, P \mid (E'_1,\cdots,E'_j;\, x_{j+1},\cdots,x_k).\, Q : \psi$ which amounts to:

$$\wedge_{i=1}^{k}\ \rho \models E_i : \vartheta_i \tag{1}$$
$$\forall V_1,\cdots,V_k : \wedge_{i=1}^{k}\ V_i \in \vartheta_i\ \Rightarrow \langle V_1,\cdots,V_k\rangle \in \kappa \tag{2}$$
$$(\rho,\kappa) \models_{\mathsf{RM}} P : \psi \tag{3}$$
$$\wedge_{i=1}^{j}\ \rho \models E'_i : \vartheta'_i \tag{4}$$
$$\forall \langle V_1,\cdots,V_k\rangle \in \kappa : \wedge_{i=1}^{j}\ V_i \in \vartheta'_i \tag{5}$$
$$\Rightarrow\ \wedge_{i=j+1}^{k}\ V_i \in \rho(\lfloor x_i \rfloor)\ \wedge\ (\rho,\kappa) \models_{\mathsf{RM}} Q : \psi$$

Furthermore we assume that $\wedge_{i=1}^{j}\ [\![E_i]\!] = [\![E'_i]\!]$ and we have to prove $(\rho,\kappa) \models P \mid Q[E_{j+1}/x_{j+1},\cdots,E_k/x_k]$. From (1) we get $\wedge_{i=1}^{k}\ \lfloor E_i \rfloor \in \vartheta_i$ since $\wedge_{i=1}^{k}\ \mathsf{fv}(E_i) = \emptyset$ and then (2) gives $\langle \lfloor E_1 \rfloor,\cdots,\lfloor E_k \rfloor\rangle \in \kappa$. From (4) and the assumption $\wedge_{i=1}^{j}\ [\![E_i]\!] = [\![E'_i]\!]$ we get $\wedge_{i=1}^{j}\ \lfloor E_i \rfloor \in \vartheta'_i$. Now (6) gives $\wedge_{i=j+1}^{k}\ \lfloor E_i \rfloor \in \rho(\lfloor x_i \rfloor)$ and $(\rho,\kappa) \models_{\mathsf{RM}} Q : \psi$. The substitution result (Lemma 3) then gives $(\rho,\kappa) \models_{\mathsf{RM}} Q[E_{j+1}/x_{j+1},\cdots,E_k/x_k] : \psi$ and together with (3) this gives the required result. The second part of the result holds trivially.

In case (Decr) we assume $(\rho,\kappa) \models \mathsf{decrypt}\ (\{E_1,\cdots,E_k\}^{\ell}_{E_0}\ [\mathsf{dest}\ \mathcal{L}])\ \mathsf{as}\ \{E'_1,\cdots,E'_j;\, x_{j+1},\cdots,x_k\}^{\ell'}_{E'_0}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P : \psi$ which amounts to:

$$\wedge_{i=0}^{k}\ \rho \models E_i : \vartheta_i \tag{6}$$
$$\forall V_0, V_1,\cdots,V_k : \wedge_{i=0}^{k}\ V_i \in \vartheta_i \tag{7}$$
$$\Rightarrow \{V_1,\cdots,V_k\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$$
$$\wedge_{i=0}^{j}\ \rho \models E'_i : \vartheta'_i \tag{8}$$
$$\forall \{V_1,\cdots,V_k\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta : \wedge_{i=0}^{j}\ V_i \in \vartheta'_i \tag{9}$$
$$\Rightarrow\ \wedge_{i=j+1}^{k}\ V_i \in \rho(\lfloor x_i \rfloor)\ \wedge$$
$$(\neg\mathsf{RM}(\ell,\mathcal{L}',\ell',\mathcal{L}) \Rightarrow (\ell,\ell') \in \psi)\ \wedge$$
$$(\rho,\kappa) \models_{\mathsf{RM}} P : \psi$$

Additionally, we assume $\wedge_{i=0}^{j}\ [\![E_i]\!] = [\![E'_i]\!]$ and we have to prove $(\rho,\kappa) \models P[E_{j+1}/x_{j+1},\cdots,E_k/x_k]$. From (6) and $\wedge_{i=0}^{k}\ \mathsf{fv}(E_i) = \emptyset$ we get $\wedge_{i=0}^{k}\ \lfloor E_i \rfloor \in \vartheta_i$ and then (7) gives $\{\lfloor E_1 \rfloor,\cdots,\lfloor E_k \rfloor\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$. From $\wedge_{i=0}^{j}\ [\![E_i]\!] = [\![E'_i]\!]$ and (8) we get $\wedge_{i=0}^{j}\ \lfloor E_i \rfloor \in \vartheta'_i$ and then (9) gives $\wedge_{i=j+1}^{k}\ \lfloor E_i \rfloor \in \rho(\lfloor x_i \rfloor)$ and $(\rho,\kappa) \models_{\mathsf{RM}} P : \psi$. Using the substitution Lemma 3 we get the required result. For the second part of the result we observe that $\neg\mathsf{RM}(\ell,\mathcal{L}',\ell',\mathcal{L}) \Rightarrow (\ell,\ell') \in \psi$ follows from (9) and since $\psi = \emptyset$ it must be the case that $\mathsf{RM}(\ell,\mathcal{L}',\ell',\mathcal{L})$. Thus the conditions of the rule (Decr) are fulfilled for $\rightarrow_{\mathsf{RM}}$.

In case (A-Decr), recall that the terms E_0 and E'_0 actually are the pair of keys m^+, m^-. Then, we proceed as for the case above, changing the additional assumption as follows: $\wedge_{i=1}^{j} \llbracket E_i \rrbracket = \llbracket E'_i \rrbracket \wedge \{E_0, E'_0\} = \{m^+, m^-\}$.

The cases (Par) and (Res) follow directly from the induction hypothesis. The case (Congr) also uses Lemma 4 about the congruence. □

Theorem 7 (Static check for reference monitor)
If $(\rho, \kappa) \models_{\mathsf{RM}} P : \emptyset$ *then* RM *cannot abort* P.

Proof. We show that there exist no Q, Q' such that $P \rightarrow^* Q \rightarrow Q'$ and $P \rightarrow^*_{\mathsf{RM}} Q \not\rightarrow_{\mathsf{RM}}$. Suppose *per absurdum* that such Q and Q' exist. Theorem 6 about subject reduction applied to $P \rightarrow^* Q$ gives $(\rho, \kappa) \models_{\mathsf{RM}} Q : \emptyset$. The generalised subject reduction result applied to $Q \rightarrow Q'$ gives $Q \rightarrow_{\mathsf{RM}} Q'$ which is a contradiction. □

Theorem 8 (Soundness of Dolev-Yao condition)
If (ρ, κ, ψ) *satisfies* $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ *of type* $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ *then* $(\rho, \kappa) \models_{\mathsf{RM}} \overline{Q} : \psi$ *for all attackers* Q *of type* $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$.

Proof. A process $\overline{Q}$ has extended type $(\{z_\bullet\}, \mathcal{N}_{\mathsf{f}} \cup \{n_\bullet, m^+_\bullet, m^-_\bullet\}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ whenever the canonical variables are in $\{z_\bullet\}$, the canonical names are in $\lfloor \mathcal{N}_{\mathsf{f}} \rfloor \cup \{n_\bullet, m^+_\bullet, m^-_\bullet\}$, all the arities used for sending or receiving are in $\mathcal{A}_\kappa$ and all the arities used for both symmetric and asymmetric encryption or decryption are in $\mathcal{A}^+_{\mathsf{Enc}}$. By structural induction on $\overline{Q}$ we prove:

- If (ρ, κ, ψ) satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ then $(\rho, \kappa) \models_{\mathsf{RM}} \overline{Q} : \psi$ for all attackers $\overline{Q}$ of extended type $(\{z_\bullet\}, \mathcal{N}_{\mathsf{f}} \cup \{n_\bullet, m^+_\bullet, m^-_\bullet\}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$.

The most interesting case is when the considered attacker $\overline{Q}$ is the process $\mathsf{decrypt}\ \overline{E}\ \mathsf{as}\ \{\overline{E_1}, \cdots, \overline{E_j}; x_{j+1}, \cdots, x_k\}^{\ell_\bullet}_{\overline{E_0}}\ [\mathsf{orig}\ \mathcal{C}]\ \mathsf{in}\ \overline{P}$ and here we need to find ϑ and $\vartheta_0, \cdots, \vartheta_j$ and show

(a) $\rho \models \overline{E} : \vartheta \wedge \wedge_{i=0}^{j} \rho \models \overline{E_i} : \vartheta_i$

and (b) for all $\{V_1, \cdots, V_k\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$ with $\wedge_{i=0}^{j} V_i \in \vartheta_i$ that:

(b_1) $\wedge_{i=j+1}^{k} V_i \in \rho(\lfloor x_i \rfloor)$

(b_2) $\neg \mathsf{RM}(\ell, \mathcal{C}, \ell_\bullet, \mathcal{L}) \Rightarrow (\ell, \ell_\bullet) \in \psi$

(b_3) $(\rho, \kappa) \models_{\mathsf{RM}} \overline{P} : \psi$

We choose ϑ as the least set such that $\rho \models \overline{E} : \vartheta$ and prove that $\vartheta \subseteq \rho(z_\bullet)$; intuitively, if $\overline{E}$ has free variables $z_1, \cdots, z_m$ then ϑ consists of all those values $\lfloor \overline{E}[V_1/z_1, \cdots, V_m/z_m] \rfloor$ where $V_i \in \rho(z_\bullet)$. We perform a similar development for $\vartheta_0, \cdots, \vartheta_j$ and this takes care of (a). Next consider $\{V_1, \cdots, V_k\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$ and assume that (c_1) $V_0 \in \vartheta_0$. Since $\vartheta_0 \subseteq \rho(z_\bullet)$, as above, we have ($\mathrm{c}_2$) $V_0 \in \rho(z_\bullet)$ and by $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ we get $V_i \in \rho(z_\bullet)$ and $\neg \mathsf{RM}(\ell, \mathcal{C}, \ell_\bullet, \mathcal{L}) \Rightarrow (\ell, \ell_\bullet) \in \psi$.

Since $\lfloor x_i \rfloor = z_\bullet$ this takes care of (b_1) and (b_2); furthermore $\overline{P}$ has extended type $(\{z_\bullet\}, \mathcal{N}_\mathsf{f} \cup \{n_\bullet, m^+_\bullet, m^-_\bullet\}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ and the induction hypothesis then takes care of (b_3).

For the case of public key decryption one follows the same argument but with $\overline{Q}$ on the form $\mathsf{decrypt}\ \overline{E}\ \mathsf{as}\ \{\!|\overline{E_1}, \cdots, \overline{E_j}; x_{j+1}, \cdots, x_k|\!\}^{\ell_\bullet}_{\overline{E_0}}\ [\mathsf{orig}\ \mathcal{C}]\ \mathsf{in}\ \overline{P}$. However, one must change (b) to become "for all $\{\!|V_1, \cdots, V_k|\!\}^{\ell}_{V_0}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$ such that $\forall (m^+, m^-) : \forall V' \in \vartheta_0 : \{V_0, V'_0\} = \{m^+, m^-\}$ and $\wedge^j_{i=1} V_i \in \vartheta_i$". Similarly, the assumption (c_1) must be changed accordingly (to become "$V'_0 \in \vartheta_0$ and $\{V_0, V'_0\} = \{m^+, m^-\}$ for some m^+, m^-") as must (c_2) (to become "$V'_0 \in \rho(z_\bullet)$").

The remaining cases are similar. □

We can now show the close connection between the Dolev-Yao condition and the notion of "hardest attackers" [55]. This result also shows the "completeness" of the Dolev-Yao condition: we have not needlessly added capabilities that cannot be possessed by real attackers:

Theorem 9 (Existence of "Hardest Attacker")
There exists an attacker Q_{hard} of type $(\mathcal{N}_\mathsf{f}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$ such that the formula $(\rho, \kappa) \models_{\mathsf{RM}} \overline{Q_{hard}} : \psi$ is equivalent to the formula $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ of type $(\mathcal{N}_\mathsf{f}, \mathcal{A}_\kappa, \mathcal{A}^+_{\mathsf{Enc}})$.

Proof. Q_{hard} is $!(|_{k \in \mathcal{A}_\kappa} Q^k_1 \mid |_{k \in \mathcal{A}^+_{\mathsf{Enc}}} Q^k_2 \mid |_{k \in \mathcal{A}^+_{\mathsf{Enc}}} Q^k_3 \mid |_{k \in \mathcal{A}_\kappa} Q^k_4 \mid Q_5 \mid |_{k \in \mathcal{A}^+_{\mathsf{Enc}}} Q^k_6 \mid |_{k \in \mathcal{A}^+_{\mathsf{Enc}}} Q^k_7 \mid Q_8)$ where Q^k_i is obtained from the i^{th} component of $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ in Table 3 for $1 \leq i \leq 5$ and in Table 9 for $6 \leq i \leq 8$. We assume that there are variables z, z_0, z_1, $\cdots$ having canonical representative $z_\bullet$ and that $1 \in \mathcal{A}_\kappa$ (as discussed in Section 5). For $\mathcal{N}_\mathsf{f} = \{n_1, \cdots, n_m\} \cup \{m^+_1, m^-_1 \cdots, m^+_p, m^-_p\}$ we then take:

$Q^k_1 = (;\, z_1, \cdots, z_k).\, \mathsf{0}$
$Q^k_2 = (;\, z).\, (;\, z_0).\, \mathsf{decrypt}\ z\ \mathsf{as}\ \{;\, z_1, \cdots, z_k\}^{\ell_\bullet}_{z_0}\ [\mathsf{orig}\ \mathcal{C}]\ \mathsf{in}\ \mathsf{0}$
$Q^k_3 = (;\, z_0). \cdots (;\, z_k).\, \langle \{z_1, \cdots, z_k\}^{\ell_\bullet}_{z_0} [\mathsf{dest}\ \mathcal{C}] \rangle.\, \mathsf{0} \mid (;\, z).\, \mathsf{0}$
$Q^k_4 = (;\, z_1). \cdots (;\, z_k).\, \langle z_1, \cdots, z_k \rangle.\, \mathsf{0}$
$Q^k_5 = \langle n_\bullet \rangle.\, \mathsf{0} \mid \langle n_1 \rangle.\, \mathsf{0} \mid \cdots \mid \langle n_m \rangle.\, \mathsf{0} \mid \langle m^+_1 \rangle.\, \mathsf{0} \mid \langle m^-_1 \rangle.\, \mathsf{0} \mid \cdots \mid \langle m^+_p \rangle.\, \mathsf{0} \mid \langle m^-_p \rangle.\, \mathsf{0} \mid (;\, z).\, \mathsf{0}$
$Q^k_6 = (;\, z).\, (;\, z_0).\, \mathsf{decrypt}\ z\ \mathsf{as}\ \{\!|;\, z_1, \cdots, z_k|\!\}^{\ell_\bullet}_{z_0}\ [\mathsf{orig}\ \mathcal{C}]\ \mathsf{in}\ \mathsf{0}$
$Q^k_7 = (;\, z_0). \cdots (;\, z_k).\, \langle \{\!|z_1, \cdots, z_k|\!\}^{\ell_\bullet}_{z_0} [\mathsf{dest}\ \mathcal{C}] \rangle.\, \mathsf{0} \mid (;\, z).\, \mathsf{0}$
$Q^k_8 = \langle m^+_\bullet \rangle.\, \mathsf{0} \mid \langle m^-_\bullet \rangle.\, \mathsf{0} \mid (;\, z).\, \mathsf{0}$ □

Theorem 10 (Authentication) *If P_{sys} guarantees static authentication then P_{sys} guarantees dynamic authentication.*

Proof. If $(\rho, \kappa) \models_{\mathsf{RM}} P_{sys} : \emptyset$ and $(\rho, \kappa, \emptyset)$ satisfies $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ then, by Theorems 7 and 8, RM does not abort $P_{sys} \mid \overline{Q}$ regardless of the choice of attacker Q. □