Automatic Validation of Protocol Narration[*]

C. Bodei[1] M. Buchholtz[2] P. Degano[1] F. Nielson[2] H. 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

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 key protocols such as Needham-Schroeder, Otway-Rees, Yahalom and Andrew Secure RPC.

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 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.

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.

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 [14] 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" [31] developed for firewall security in Mobile Ambients.

We then address a form of authenticity by statically verifying the origin and destination of messages [18]. More specifically, we verify whether a message encrypted by A and intended for B does indeed come from A and reaches B only. This suffices for dealing with authenticity problems in the protocols mentioned above, in particular with our run-

[*]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-projects AI, TS & CFA, and MEFISTO.

ning example, the Wide Mouthed Frog protocol. Because of the approximative nature of static analysis 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 we briefly outline (in Section 6) our polynomial-time implementation using tree grammars. This follows the approach taken in [32] 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 [33]; 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 demonstrate (in Section 7) that our technique is strong enough to report the known problems in symmetric key protocols such as Needham-Schroeder [28, 29], Otway-Rees [36], Yahalom [10] and Andrew Secure RPC [40]. Furthermore, it is strong enough that it does not report flaws in suitably amended versions of these protocols.

We conclude (in Section 8) with an assessment of our approach and its ability to deal with related security notions. Appendix A summarises the protocol narrations considered, Appendix B contains proofs of our main results.

Comparison with state of the art. A number of logical theories have been developed for confidentiality [9, 14, 37] based on perfect cryptography; they have often been investigated with the help of semi-automatic theorem provers resulting in a rather high computational overhead. A related approach is that of the NRL protocol analyser [24] where (possibly confluent) reduction systems are used to mechanically handle terms involving explicit cryptographic primitives, like exponentiation. Also logical theories have been used for authenticity [10, 19, 5, 43] based on perfect cryptography; these specifications are not mechanised and sometimes the resulting formulae lose the operational flavour of protocols thereby making it hard to establish soundness with respect to an operational semantics.

To get a more operational view many papers have considered process algebras for expressing protocols and analysis methods have been applied accordingly. For example, type (and effect) systems retain some of the flavour of logic based approaches but facilitate reasoning at a level closer to the syntactic presentation of the protocol [4, 1, 3, 20, 21]; mostly they focus on type checking (usually polynomial-time decidable) rather than type inference (appearing to be computationally intractable due to the absence of principal types). Our approach is more in the flavour of type inference while retaining the polynomial-time complexity. Other semantic methods are based on testing equivalence or bisimulation to provide the semantics correctness of the approach; e.g. [17, 15] follow this line to directly verify protocols in an operational setting.

The model checking approach is also oriented towards an operational approach, e.g. *Interrogator* [26] and *murϕ* [27]. Here a finite state automaton is built to represent the behaviour of the protocol, occasionally modelling causality [16], and an exhaustive search is performed to verify that each reachable state enjoys the desired modal formula. Lowe [22] specified protocols in CSP and exploited the operational semantics to construct their (finite) models thereby discovering 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 [39, 41].

Language based approaches are built around a detailed high level language for expressing protocols, e.g. CAPSL [12] where the resulting programs are translated into a Horn fragment of linear logic and proofs of security properties utilise tools like inductive verifiers, model checkers, or a PROLOG based constraint solver [25].

Static analysis based on control flow analysis has shown promise of analysing such systems [7, 8, 6, 31] but has so far not been able to demonstrate its ability to find flaws in protocols. In this paper we demonstrate the feasibility of applying static analysis to finding flaws in protocol. The main strong point of our approach is the ability to perform a fully automatic analysis.

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 [14] – 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 [4] of the Wide Mouthed Frog protocol [10] (abbreviated WMF) aiming at establishing a secret session key K between the two principals A and B sharing master keys K_A resp. K_B with a trusted server S:

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

Usually protocols narrations come along with additional explanations. 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 sender 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 and we are explicit on checks 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: $\mathit{source}, \mathit{destination}, \mathit{message}_1, \cdots, \mathit{message}_k$ followed by assumptions or checks in square brackets:

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

The first line describes the actions of the sender of the message while the next two lines describe 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 $[\text{orig } x_A]$.

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

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

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 $[\text{orig } S]$ in line $2''$). Other properties, e.g. confidentiality or freshness of various data, can be addressed in the same style (see the Conclusion).

Note that the two steps leading to an extended protocol narration are systematic. This approach is similar to the one taken by Casper [23], CAPSL [12, 11], and CVS [15].

3. The LySa-calculus

The considerations of Section 2 motivate defining a new process algebra, LySa. It differs from the π- and Spi-calculus 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. The second difference of LySa is that the tests associated with input and decryption are naturally expressed using pattern matching.

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.

[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 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 [4]:

$P ::=$	*processes*
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 0 of processes.)

For ease of presentation we restrict ourselves to a very simple form of patterns; whenever matching a k-tuple of values we allow to match on a *prefix* of $0 \leq j \leq k$ values and then to bind the remaining $k - j$ values to variables. Syntactically, this is indicated by using a semi-colon to separate the components where matching is performed from those where only binding takes place. Both input and decryption only succeed on values where the match succeeds. Hence, we dispense with an explicit matching construct.

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.

Similarly, we can deal with both perfect hash functions and perfect message authentication codes. In the former case we encrypt messages simply using a name H meant only for hash functions; in the latter case we encrypt messages using a pair (MAC, K) where MAC is a name used only for message authentication codes and K is a symmetric key. In both cases we make sure that the corresponding decryptions *do not bind any variables* (after the semi-colon); in this way decryption already serves as the required test on hash values. We can also deal with history-dependent encryption in the same style of [8].

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 omit $[\mathsf{dest}\ \mathcal{L}]$ and $[\mathsf{orig}\ \mathcal{L}]$ in case $\mathcal{L}$ is $\mathcal{C}$ and we write $[\mathsf{dest}\ \ell]$ and $[\mathsf{orig}\ \ell]$ instead of the more cumbersome $[\mathsf{dest}\ \{\ell\}]$ and $[\mathsf{orig}\ \{\ell\}]$.

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

To simplify the definition of the control flow analysis in Section 4, we discipline the α-renaming of bound names. We assume 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 remark 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.

Semantics. Following the tradition of the π-calculus, we shall give LySa a reduction semantics. The *reduction relation* $\rightarrow_{\mathcal{R}}$ is the least relation on closed processes that satisfies the rules in Table 1. It uses the standard notion of substitution, $P[E/x]$, and structural congruence, $\equiv$, apart from the disciplined treatment of α-conversion, discussed above. 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 $\mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) = (\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L})$;

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

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

$$\text{(Par)}\ \frac{P \rightarrow_{\mathcal{R}} P'}{P \mid Q \rightarrow_{\mathcal{R}} P' \mid Q} \qquad \text{(Res)}\ \frac{P \rightarrow_{\mathcal{R}} P'}{(\nu\, n)P \rightarrow_{\mathcal{R}} (\nu\, n)P'} \qquad \text{(Congr)}\ \frac{P \equiv Q \ \wedge\ Q \rightarrow_{\mathcal{R}} Q' \ \wedge\ Q' \equiv P'}{P \rightarrow_{\mathcal{R}} P'}$$

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

- 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 $[\![\cdot]\!]$. When the matchings are successful each E_i is bound to each x_i.

Similarly, the rule (Decr) expresses the result of matching an encryption $\{E_1, \cdots, E_j, E_{j+1}, \cdots, E_k\}_{E_0}^{\ell}[\mathsf{dest}\ \mathcal{L}]$ with $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E'_1, \cdots, E'_j; x_{j+1}, \cdots, x_k\}_{E'_0}^{\ell'}\ [\mathsf{orig}\ \mathcal{L}']$ $\mathsf{in}\ P$, i.e. with a corresponding decryption. As 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 as well as public key cryptography.

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. sender (A), receiver (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 *one* server, which is modelled in the same way: its name is $I_{\text{-}1}$ and it will serve in the role of 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:

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

Here the checks of the extended narration above are performed by matching on inputs and decryptions[2]. The separation of different branches of the matching allows us to keep the analysis simple and efficient.

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_i^A and K_i^B are used for the two roles of the principals.

[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.

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. (A principal does not want to authenticate itself; hence $j \neq i$. If needed, e.g. for checking confidentiality, we can remove this constraint.) 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 $\{S\}$). 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 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 (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 give a safe approximation to when the reference monitor may abort the computation.

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 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 $\rho \models E : \{\lfloor E \rfloor\}$. Furthermore:

- $\rho \models E : \vartheta$ and $\lfloor E' \rfloor \in \rho(\lfloor x \rfloor)$ imply $\rho \models E[E'/x] : \vartheta$.

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 $V \mathrel{\sqsubseteq} \vartheta$ that holds if there is a value V' in ϑ that equals V when the annotations are ignored, formally:

$$V \mathrel{\sqsubseteq} \vartheta \quad \text{iff} \quad \exists V' \in \vartheta : [\![V]\!] = [\![V']\!]$$

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.

To obtain this information we shall, as for terms, make use of the abstract environment ρ. 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 2 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. This 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 *input* the terms $E_1, \cdots, E_j$ are used for matching values sent on the network. Thus, the rule for input (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 $V_i \mathrel{\sqsubseteq} \vartheta_i$, 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 $\{V_0, \cdots, V_k\}_{V_0}^{\ell'}[\mathsf{dest}\ \mathcal{L}'] \in$

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

$$(\rho, \kappa) \models_{\mathsf{RM}} 0 : \psi$$

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

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

$$\frac{\begin{array}{c}\rho \models E : \vartheta \;\;\wedge\;\; \wedge_{i=0}^{j}\, \rho \models E_i : \vartheta_i \,\wedge \\ \forall\ \{V_1, \cdots, V_k\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] \in \vartheta :\; \wedge_{i=0}^{j}\, V_i \mathrel{\mathsf{E}} \vartheta_i \;\Rightarrow\; \begin{array}[t]{l}\wedge_{i=j+1}^{k}\, V_i \in \rho(\lfloor x_i\rfloor) \,\wedge \\ (\neg\mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell, \ell') \in \psi) \,\wedge \\ (\rho, \kappa) \models_{\mathsf{RM}} P : \psi\end{array}\end{array}}{(\rho, \kappa) \models_{\mathsf{RM}} \mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}^{\ell'}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P : \psi}$$

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

ϑ whether the values $V_1, \ldots V_j$ 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 is updated to 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 have the following results:

- $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ and $\lfloor E'\rfloor \in \rho(\lfloor x\rfloor)$ imply $(\rho, \kappa) \models_{\mathsf{RM}} P[E'/x] : \psi$.
- If $P \equiv Q$ then $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ iff $(\rho, \kappa) \models_{\mathsf{RM}} Q : \psi$.

The subject reduction result below expresses that our analysis is semantically correct regardless of the way the semantics is parameterised:

Theorem 1 *If $P \rightarrow_{\mathcal{R}} Q$ and $(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$ then also $(\rho, \kappa) \models_{\mathsf{RM}} Q : \psi$; in particular this holds for the standard semantics as well as the reference monitor semantics.*

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 *If $(\rho, \kappa) \models_{\mathsf{RM}} P : \emptyset$ then* RM *cannot abort P.*

Analysis of WMF. For the protocol of Section 2 we have

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

where ρ has these non-empty entries (for $1 \leq i, j \leq n$, $i \neq j$, and $1 \leq l \leq k$)

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

and κ is:

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

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}; 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 if (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 [14] 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 [31]. We base our presentation on the first approach and then show the close connection between the two approaches in Theorem 4.

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 $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$; to avoid having to deal with too many special cases we shall assume that $\mathcal{A}_\kappa$ contains at least one positive integer (e.g. 1). We claim that there is no loss of generality in assuming that attackers have type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ as well; in particular, we claim that the ability of the attacker to use a "private channel" based on k-ary communication for $k \notin \mathcal{A}_\kappa$ or k-ary cryptography for $k \notin \mathcal{A}_{\mathsf{Enc}}$, does not increase its computational power.

One aspect 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 $(E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k).\, P$ and $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}^{\ell}_{E_0}\ [\mathsf{orig}\ \mathcal{L}]\ \mathsf{in}\ P$ 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, our syntax requires annotations 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 $\overline{Q}$ 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 $\overline{Q}$ 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.

We can now establish the soundness of the Dolev-Yao condition for LySa:

Theorem 3 *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" [31]. 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 *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 A_i (written $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}^{\ell}_{E_0}\ [\mathsf{orig}\ A_i]\ \mathsf{in}\ P$) 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.

(1) $\wedge_{k\in\mathcal{A}_\kappa}\ \forall\langle V_1,\cdots,V_k\rangle\in\kappa:\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)$	the attacker may learn by eavesdropping
(2) $\wedge_{k\in\mathcal{A}_{\mathsf{Enc}}}\ \forall\{V_1,\cdots,V_k\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}]\in\rho(z_\bullet):$ $V_0\ \mathsf{E}\ \rho(z_\bullet)\Rightarrow(\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\ \wedge$ $(\neg\mathsf{RM}(\ell,\mathcal{C},\ell_\bullet,\mathcal{L})\Rightarrow(\ell,\ell_\bullet)\in\psi))$	the attacker may learn by decrypting messages with keys already known
(3) $\wedge_{k\in\mathcal{A}_{\mathsf{Enc}}}\ \forall V_0,\cdots,V_k:\wedge_{i=0}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow$ $\{V_1,\cdots,V_k\}_{V_0}^{\ell_\bullet}[\mathsf{dest}\ \mathcal{C}]\in\rho(z_\bullet)$	the attacker may construct new encryptions using the keys known
(4) $\wedge_{k\in\mathcal{A}_\kappa}\ \forall V_1,\cdots,V_k:\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow\langle V_1,\cdots,V_k\rangle\in\kappa$	the attacker may actively forge new communications
(5) $\{n_\bullet\}\cup\lfloor\mathcal{N}_{\mathsf{f}}\rfloor\subseteq\rho(z_\bullet)$	the attacker initially has some knowledge

Table 3. Dolev-Yao condition.

$A\to S:\ A,\{B,K\}_{K_A}$	$M\to S:\ M,\{B,K\}_{K_M}$	$A\to M_S:\ A,B,\{K\}_{K_A}$	$A\to M_S:A,B,\{K\}_{K_A}$	$A\to M_S:A,B,\{K\}_{K_A}$
$S\to M_B:A,\{K\}_{K_B}$	$S\to M_B:M,\{K\}_{K_B}$	$M_A\to S:\ A,B',\{K\}_{K_A}$	$M_S\to S:\ A,M,\{K\}_{K_A}$	$M_S\to S:A,M,\{K\}_{K_A}$
$M_S\to B:A',\{K\}_{K_B}$	$M_S\to B:A',\{K\}_{K_B}$	$S\to B':\ \{A,K\}_{K_{B'}}$	$S\to M:\ \{A,K\}_{K_M}$	$S\to M:\ \{A,K\}_{K_M}$
$A\to B:\ \{m_1\cdots m_k\}_K$	$M\to B:\ \{m_1\cdots m_k\}_K$	$A\to M_B:\ \{m_1\cdots m_k\}_K$	$A\to M_B:\{m_1\cdots m_k\}_K$	$M_A\to S:A,B,\{K\}_{K_A}$
		$M_A\to B':\{m_1\cdots m_k\}_K$		$S\to B:\ \{A,K\}_{K_B}$
				$M\to B:\ \{m_1\cdots m_k\}_K$
Attack 1	Attack 2	Attack 3	Attack 4	Attack 5

Table 4. Attacks on WMF **variations.**

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 $(\rho,\kappa)\models_{\mathsf{RM}}P:\emptyset$ and $(\rho,\kappa,\emptyset)$ satisfies $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$.

Theorem 5 *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 $\mathcal{F}_{\underline{\mathsf{RM}}}^{\mathsf{DY}}$ then, by Theorems 2 and 3, RM does not abort $P_{sys}\mid\overline{Q}$ regardless of the choice of attacker Q. □

Validation of WMF. We analyse the WMF protocol of Section 2 and restrict our attention to the solutions that satisfy the formula $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$ 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 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 $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$. To obtain an implementation we use the Succinct Solver [33], which for a formula of an extensions of Horn clauses, called Alternation-free Least Fixed Point logic (ALFP), finds the least interpretation of the predicate symbols in the formula, which satisfies the formula. The logic is interpreted over a *finite universe* unlike the components ρ,κ and ϑ, which contain elements from the infinite universe of terms. To obtain an efficient implementation we transform the analysis into a logically equivalent formulation written in ALFP and we also ensure that the terms can be encoded over a finite universe. We proceed as follows.

Firstly, the specification of the analysis in Table 2 is succinct [35]; using the techniques of [34] it is transformed into a verbose specification [35]. 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, applying techniques from [32] the specification is transformed to use a *finite universe* by encoding terms as production rules in a tree grammar. Essentially, this boils down to representing encrypted terms by one level of nesting. Remember that we have added labels (written as superscripts and for now ignoring annotations) so a term such as $\{E^{l_2}\}_{K^{l_1}}$ will be represented by the value $\{l_2\}_{l_1}$. Assuming that K is a name we then keep a production rule $l_1 \rightarrow K$ along with production rules for the non-terminal l_2 describing the value of the term E. This lets us represent infinitely many values by a finite number of production rules. Take for example the process

$$\langle n^{l_1}\rangle \mid !(x).\langle\{x^{l_4}\}^{l_2}_{K^{l_3}}\rangle$$

which encrypts the name n arbitrarily many times under the key K. We represent all the possible values occurring by the production rules $l_1 \rightarrow n, l_2 \rightarrow \{l_4\}_{l_3}, l_3 \rightarrow K$, and $l_4 \rightarrow l_1$ and $l_4 \rightarrow l_2$. Note in particular that the applied occurrence of x at l_4 may evaluate to both the value from l_1 (i.e. to n) and to all the values of the encrypted term at l_2. The latter gives rise to a cycle in the grammar, thus, representing encryptions of arbitrary nesting depth. The simple structure of the terms allows us in general to represent all possible values, which occur during process execution, by a *finite* number of production rules.

Thirdly, we transform 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.

The formula $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ for the attacker is transformed in a similar way as described in the second and third step above. Finally, the analysis is turned into a formula generation function that given a process P produces an ALFP formula, which represents the analysis of P. This formula together with the transformed $\mathcal{F}^{\mathsf{DY}}_{\mathsf{RM}}$ is then given to the Succinct Solver, which calculates the values of the analysis components ρ, κ, and ψ (which by now are represented in a different, but equivalent, way to those used in Table 2).

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 [33]. For our current implementation the nesting depth is governed by the maximal length of the sequences used in communication and encryption though techniques from [32] might have been be applied to yield a cubic worst-case upper bound. In practice, the implementation runs in sub-cubic time and we obtain running times well under one minute for all the experiments conducted in the next section.

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) [4, 10], Needham-Schroeder [28], Amended Needham-Schroeder [29], Otway-Rees [36], Yahalom [10] and Andrew Secure RPC [40]. 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 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).

When roles as well as master keys are kept distinct, as shown in the first column of Table 5, 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. [15]) 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 [37], which exploits that encrypted messages from a principal acting both as initiator and responder may be confused.

In [10] an optimised version of the Yahalom protocol is suggested and an attack is reported in [42]; from Table 5

protocol $1 \le i, j \le n, i \neq j$	$A \neq B$ $\wedge_{i=0}^{n} K_i^A \neq K_i^B$	$A = B$ $\wedge_{i=0}^{n} K_i^A \neq K_i^B$	$A \neq B$ $\wedge_{i=0}^{n} K_i^A = K_i^B$	$A = B$ $\wedge_{i=0}^{n} K_i^A = K_i^B$
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.

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

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

we see that the attack only succeeds when not distinguishing between roles and when using the same master key for distinct roles. Paulson [38] suggests an amendment whose correctness we can 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 (see Section 2) 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 [4] is problematic when old session keys are leaked. The original presentation in [10] 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 [13]. 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 [38], 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 [30]. 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 [10]; our analysis confirms that the amended version suggested in [10] indeed solves the problem.

8. 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.

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 and 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. Since 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 the diligent use of Abstract Syntax Notation One (ASN.1) will provide the necessary safe guards. In subsequent work we plan to deal with perfect asymmetric cryptography, and a more direct treatment of hash functions and message authentication codes so that we can also account for 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 Woo and Lam's idea of correspondence assertions [44].

Our techniques are also able to deal with a number of other security properties, e.g. secrecy. The present development allows us 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 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 [44].

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 their Properties. In *Foundations of Secure Computation.* NATO Science Series, IOS Press (2000), 39-60.

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

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

[5] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proc. Symposium on Principles of Distributed Computing*, pp. 201–216. ACM Press, 1991.

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

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

[8] C. Bodei, P. Degano, F. Nielson, and H. R. Nielson. Flow logic for Dolev-Yao secrecy in cryptographic processes. *FGCS*, 18(6):747–756, 2002.

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

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

[11] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A Meta-notation for Protocol Analysis, In *Proc. CSFW*, 1999.

[12] G. Denker and J. Millen. CAPSL integrated protocol environment. In *DARPA Information Survivability Conference*, pp. 207–221. IEEE Computer Society, 2000.

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

[14] D. Dolev and A. Yao. On the security of public key protocols. *IEEE TIT*, IT-29(12):198–208, 1983.

[15] A. Durante, R. Focardi, and R. Gorrieri. A compiler for analysing cryptographic protocols using non-interference. *ACM ToSEM*, 9(4):488–528, 2000.

[16] F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. Conf. on Security and Privacy*, pp. 160–171, 1998. IEEE Press.

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

[18] D. Gollmann. What do we mean by Entity Authentication. In *Proc. Symposium on Security and Privacy*, pp. 46–54. IEEE Computer Society Press, 1996.

[19] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proc. Symposium on Research in Security and Privacy*, pp. 234–248. IEEE Computer Society Press, 1990.

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

[21] A. D. Gordon and A. Jeffrey. Types and Effects for Asymmetric Cryptographic Protocols. In *Proc. CSFW*, 2002.

[22] G. Lowe. An attack on the Needham-Schroeder public-key authentification protocol. *IPL*, 56(3):131–133, 1995.

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

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

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

[26] J. K. Millen. The Interrogator: A tool for cryptographic protocol security. In *Proc. Symposium on Security and Privacy*, pp. 134–141. IEEE Computer Society Press, 1984.

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

[28] 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.

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

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

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

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

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

[34] H. R. Nielson and F. Nielson. Flow logics for constraint based analysis. In *Proc. CC'98*, LNCS 1383, pp. 109–127. Springer, 1998.

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

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

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

[38] L. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. TR, Cambridge University, England, 1998.

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

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

[41] S. Schneider. Security properties and CSP. In *Proc. Symposium on Research in Security and Privacy*, pp. 174–187. IEEE Computer Society Press, 1996.

[42] P. Syverson. A taxonomy of replay attacks. In *Proc. CSFW*, pp. 187–191. IEEE, 1994.

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

[44] T. Y. C. Woo and S. S. Lam. A Semantic Model for Authentification Protocols in em Proc. of the IEEE Symposium on Security and Privacy, pp. 178–195, 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. [4]

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

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. $A \to B: \quad \{m_1, \cdots, m_k\}_K$

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

Needham-Schroeder (symmetric key). [28]

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. [29]

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. [36]

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. [10]

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 [38]:

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. [40]

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.

B. Proofs

Substitution result.

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

Both proofs are by a straightforward structural induction.

Congruence result.

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

The proof amounts to a straightforward inspection of each of the clauses defining $P \equiv Q$.

Theorem 1: Subject reduction theorem. We prove the more general result

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

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

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:

$$\wedge_{i=1}^{k} \rho \models E_i : \vartheta_i \tag{1}$$

$$\forall V_1, \cdots, V_k : \wedge_{i=1}^{k} V_i \in \vartheta_i \Rightarrow \langle V_1, \cdots, V_k \rangle \in \kappa \tag{2}$$

$$(\rho, \kappa) \models_{\mathsf{RM}} P : \psi \tag{3}$$

$$\wedge_{i=1}^{j} \rho \models E'_i : \vartheta'_i \tag{4}$$

$$\begin{aligned} &\forall \langle V_1, \cdots, V_k \rangle \in \kappa : \wedge_{i=1}^{j} V_i \mathrel{\mathsf{E}} \vartheta'_i \\ &\quad \Rightarrow \wedge_{i=j+1}^{k} V_i \in \rho(\lfloor x_i \rfloor) \wedge (\rho, \kappa) \models_{\mathsf{RM}} Q : \psi \end{aligned} \tag{5}$$

Furthermore we assume that $\wedge_{i=1}^{j} \llbracket E_i \rrbracket = \llbracket E'_i \rrbracket$ 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 $\wedge_{i=1}^{j} \llbracket E_i \rrbracket = \llbracket E'_i \rrbracket$ we get $\wedge_{i=1}^{j} \lfloor E_i \rfloor \mathrel{\mathsf{E}} \vartheta'_i$. 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 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 $(\rho, \kappa) \models \mathsf{decrypt}\ (\{E_1, \cdots, E_k\}_{E_0}^{\ell}\ [\mathsf{dest}\ \mathcal{L}])\ \mathsf{as}\ \{E'_1, \cdots, E'_j; x_{j+1}, \cdots, x_k\}_{E'_0}^{\ell'}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P : \psi$

which amounts to:

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

Furthermore we assume $\wedge_{i=0}^{j} \llbracket E_i \rrbracket = \llbracket E_i' \rrbracket$ 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 $\{\lfloor E_1 \rfloor, \cdots, \lfloor E_k \rfloor\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$. From $\wedge_{i=0}^{j} \llbracket E_i \rrbracket = \llbracket E_i' \rrbracket$ and (8) we get $\wedge_{i=0}^{j} \lfloor E_i \rfloor \mathrel{\mathsf{E}} \vartheta_i'$ 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 result 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 $\rightarrow_{\mathsf{RM}}$.

The cases (Par) and (Res) follow directly from the induction hypothesis. The case (Congr) also uses the congruence result.

Theorem 2: Static check for reference monitor.

- Assume $(\rho, \kappa) \models_{\mathsf{RM}} P : \emptyset$. Then, there exist no Q, Q' such that $P \rightarrow^* Q \rightarrow Q'$ and $P \rightarrow_{\mathsf{RM}}^* Q \not\rightarrow_{\mathsf{RM}}$.

To prove this suppose *per absurdum* that such Q and Q' exist. The subject reduction result 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 3: Correctness of Dolev-Yao condition. A process $\overline{Q}$ has extended type $(\{z_\bullet\}, \mathcal{N}_{\mathsf{f}} \cup \{n_\bullet\}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ whenever the canonical variables are in $\{z_\bullet\}$, the canonical names are in $\lfloor \mathcal{N}_{\mathsf{f}} \rfloor \cup \{n_\bullet\}$, all the arities used for sending or receiving are in $\mathcal{A}_\kappa$ and all the arities used for encryption or decryption are in $\mathcal{A}_{\mathsf{Enc}}$. By structural induction on $\overline{Q}$ (see Section 5) we prove:

- If (ρ, κ, ψ) satisfies $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$ 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 $(\{z_\bullet\}, \mathcal{N}_{\mathsf{f}} \cup \{n_\bullet\}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$.

The most interesting case is when $\overline{Q}$ is the process $\mathsf{decrypt}\ \overline{E}\ \mathsf{as}\ \{\overline{E_1}, \cdots, \overline{E_j};\ x_{j+1}, \cdots, x_k\}_{\overline{E_0}}^{\ell_\bullet}\ [\mathsf{orig}\ \mathcal{C}]\ \mathsf{in}\ \overline{P}$ 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 for all $\{V_1, \cdots, V_k\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$ with $\wedge_{i=0}^{j} V_i \mathrel{\mathsf{E}} \vartheta_i$ that:

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

(c) $\neg \mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell, \ell') \in \psi$

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

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 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 $\{V_1, \cdots, V_k\}_{V_0}^{\ell}[\mathsf{dest}\ \mathcal{L}] \in \vartheta$ and assume that $V_0 \mathrel{\mathsf{E}} \vartheta_0$. Since $\vartheta_0 \subseteq \rho(z_\bullet)$, as above, we have $V_0 \mathrel{\mathsf{E}} \rho(z_\bullet)$ and by $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$ we get $V_i \in \rho(z_\bullet)$ and $\neg \mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell, \ell') \in \psi$. Since $\lfloor x_i \rfloor = z_\bullet$ this takes care of (b) and (c); furthermore $\overline{P}$ has type $(\{z_\bullet\}, \mathcal{N}_{\mathsf{f}} \cup \{n_\bullet\}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$ and the induction hypothesis then takes care of (d).

The remaining cases are similar.

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 $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$ of type $(\mathcal{N}_{\mathsf{f}}, \mathcal{A}_\kappa, \mathcal{A}_{\mathsf{Enc}})$.

Q_{hard} is $!\,(|_{k \in \mathcal{A}_\kappa} Q_1^k \mid |_{k \in \mathcal{A}_{\mathsf{Enc}}} Q_2^k \mid |_{k \in \mathcal{A}_{\mathsf{Enc}}} Q_3^k \mid |_{k \in \mathcal{A}_\kappa} Q_4^k \mid Q_5)$ where Q_i^k is obtained from the i'th component of $\mathcal{F}_{\mathsf{RM}}^{\mathsf{DY}}$. 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) and (for $\mathcal{N}_{\mathsf{f}} = \{n_1, \cdots, n_m\}$) we then take:

$$Q_1^k = (;\, z_1, \cdots, z_k).\, 0$$
$$Q_2^k = (;\, z).\, (;\, z_0).\ \mathsf{decrypt}\ z\ \mathsf{as}\ \{;\, z_1, \cdots, z_k\}_{z_0}^{\ell_\bullet}\ [\mathsf{orig}\ \mathcal{C}]\ \mathsf{in}\ 0$$
$$Q_3^k = ((;\, z_0). \cdots (;\, z_k).\, \{z_1, \cdots, z_k\}_{z_0}^{\ell_\bullet}[\mathsf{dest}\ \mathcal{C}]) \mid (;\, z).\, 0$$
$$Q_4^k = (;\, z_1). \cdots (;\, z_k).\, \langle z_1, \cdots, z_k \rangle.\, 0$$
$$Q_5^k = \langle n_\bullet \rangle.\, 0 \mid \langle n_1 \rangle.\, 0 \mid \cdots \mid \langle n_m \rangle.\, 0 \mid (;\, z).\, 0$$