Analyzing Security Protocols in Hierarchical Networks

Ye Zhang and Hanne Riis Nielson

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

Abstract. Validating security protocols is a well-known hard problem even in a simple setting of a single global network. But a real network often consists of, besides the public-accessed part, several sub-networks and thereby forms a hierarchical structure. In this paper we first present a process calculus capturing the characteristics of hierarchical networks and describe the behavior of protocols on such networks. We then develop a static analysis to automate the validation. Finally we demonstrate how the technique can benefit the protocol development and the design of network systems by presenting a series of experiments we have conducted.

1 Introduction

With the fast development of the communication technology, thousands of intranets of companies, colleges, etc. are connected via the Internet. The network structure may even change dynamically as exemplified when relocating a laptop from one place to another. Consider the example on the left of Figure 1 where gateways are inserted between local networks so that the locally exchanged messages are not available outside. A tree that represents the network structure is presented on the right of the figure; here the internal nodes denote the networks and the leaves represent the agents. The network hierarchy, therefore, requires that all messages sent between the server and the laptop must go through the office network.

The fact that the communication varies from place to place increases the complexity of protocol analysis. Also such networks present us with a new challenge of defining the attacker capabilities since the classical Dolev-Yao model [9] was originally proposed by assuming the existence of a single global network, the Internet. In this paper we shall present our approach to deal with these issues.

Overview of the Paper. In Section 2 we present a variant of the Ambient calculus [7, 4, 5] to model hierarchical networks as well as security protocols; in order to formalize authentication properties we syntactically add annotations for declaring authentication intentions of the protocol. In Section 3 we develop a control flow analysis [15, 18] for tracking the interested property. Regarding the communication environment considered in this paper, we declare the attacker

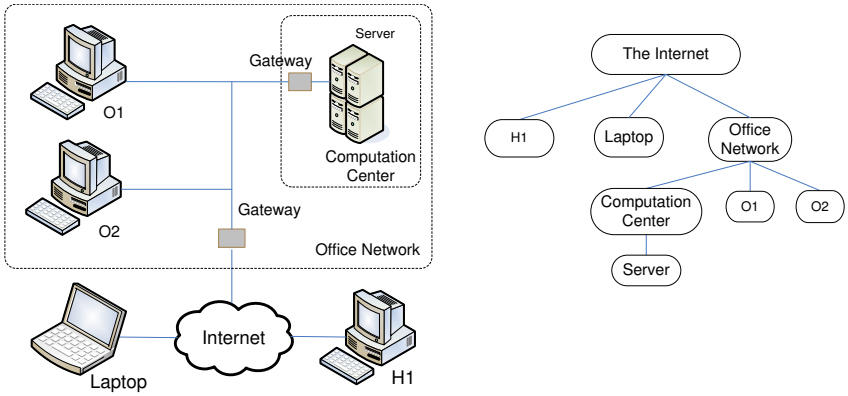


Fig. 1. Hierarchical network: an example.

capability based on the Dolev-Yao conditions in Section 4. Our analysis is fully automatic and always terminating; in Section 5 we sketch the implementation and show its running-time is polynomial in the size of ambient processes. Section 6 reports our experimental results on a series of virtual networks and protocols. Finally we conclude with a brief assessment of our approach and a comparison with related work in Section 7.

2 ABoxed Ambients

We base ourselves on Boxed Ambients [4] and customize it in several ways. First we remove nil capability ϵ and concatenation $M_1.M_2$ from Boxed Ambients. Then we extend the calculus with annotations for specifying the authentication intentions of protocols explicitly. Finally our calculus deviates from all other ambient calculi, e.g. Mobile Ambients [7] and Discretionary Ambients [18], in having attacker processes that are used to declare the locations accessible to attackers.

The syntax of processes P, communication directions η, and capabilities M is given by Table 1. While most constructs are standard, the further explanation goes to the restriction and input primitives. The two restriction constructs have same effect on all processes except for attacker processes: suppose an attacker is inside P, restriction $(v\ n)P$ allows the value n to be part of initial knowledge of the attacker while secret restriction $(vk\ n)P$ keeps the value unknown to the attacker. For the simplicity of the presentation, we assume that a subset $\mathbf{C} \subseteq \mathbf{Name}$ of names is kept for constants and demand that the name introduced by two restriction constructs are constants. For input constructs we, inspired by Lysa [2], use a simple form of patterns, $(M'_1, \cdots, M'_{\mathrm{j}}; x_{j+1}, \cdots, x_k)^\eta$, to be matched against a k-tuple of values $(M_1, \cdots, M_{\mathrm{k}})$. The idea is that the matching succeeds if the first $1 \leq i \leq j$ values M'_i pairwise correspond to the values M_i; if

$P ::=$	$(v\ n)P$	restriction	$\eta ::= n$	child
$\mid$	$(vk\ n)P$	secretrestriction	$\mid \uparrow$	parent
$\mid$	0	inactiveprocess	$\mid \circ$	local
$\mid$	$P_1 \mid P_2$	composition		
$\mid$	$!P$	replication	$M ::=$ in n	enter n
$\mid$	$n[P]$	ambient	$\mid$ out n	exit n
$\mid$	$M.P$	movement	$\mid n$	name
$\mid$	$\langle M_1, \cdots, M_{\mathrm{k}} \rangle^{\eta}$	output		
$\mid$	$(M_1, \cdots, M_{\mathrm{j}}; x_{j+1}, \cdots, x_k)^{\eta}.P$	input		
$\mid$	$\bullet$	attacker		

Table 1. Syntax of ABoxed Ambients.

so, the remaining $k-j$ values are bound to the variables $x_{j+1}, \cdots, x_k$ respectively. For the sake of simplicity, we shall enforce that $x_i \in \mathbf{V}$ where $\mathbf{V} = \mathbf{Name} \setminus \mathbf{C}$.

We assume perfect cryptography in this paper and make use of two processes, local-output and input-from-child, to model encryption and decryption respectively. The intuition is that in order to read the mailbox of a child a parent must have known his child's name (encryption key). To check protocol intentions, we syntactically annotate the pair by:

$$\langle M_1, \cdots, M_{\mathrm{k}} \rangle^{\circ}_{\ell}[\text{dest } \mathcal{L}]$$
$$(M_1^{'}, \cdots, M_{\mathrm{j}}^{'}; x_{\mathrm{j}+1}, \cdots, x_{\mathrm{k}})^{n}_{\ell}[\text{orig } \mathcal{L}]$$

where label ℓ (called crypt-point) is from some enumerable set $\mathcal{D}$ disjoint from **Name** and is added to program points where encryption and decryption happen. The assertion [dest $\mathcal{L}$] specifies a set of crypt-points $\mathcal{L} \subseteq \mathcal{D}$ where the message is intended to be decrypted. Similarly [orig $\mathcal{L}$] lists all desired crypt-points at which M is allowed to have been encrypted. A more detailed discussion on how to encode encryption and decryption with ambient calculus can be found in [19].

To simplify the analysis definition in Section 3, we shall suppose that each name has a *canonical name* $\lfloor n \rfloor \in \mathbf{Name}$ and require the alpha-renaming preserves the canonical name; therefore only the canonical version of a name will be recorded in the analysis. Similarly we write $\lfloor M \rfloor$ for the *canonical capability* of M where the name or variable is replaced with its canonical version. To formulate protocols and networks more precisely and get better analysis results, we classify ambients into two classes: site ambients which formalize local networks and computers, and packet ambients which describe data objects moving between sites. The programs of interest are then ambients in the form of $n_\star[P_\star]$ where $n_\star \notin \mathrm{fn}(\mathrm{P}_\star)$ and the function $\mathrm{fn}(P_\star)$ collects the free names of $P_\star$. Formally $P_\star$ satisfies the conjunction of the following conditions:

- any free name of $P_\star$ is from $\mathbf{C}$; formally $\lfloor \mathrm{fn}(\mathrm{P}_\star) \rfloor \subseteq \mathbf{C}$;
- $P_\star$ is well-formed with respect to $\mathbf{C}$; formally $\mathbf{C} \vdash wf_{\,s}(P_\star)$.

Here the canonicity operation $\lfloor \cdot \rfloor$ is extended in a pointwise manner; the well-formedness basically demands: (1) sites are not movable and thereby the network

Alpha − renaming :	Replication :
$P \equiv Q$ if P are disciplined $\alpha-$equivalent to Q	$!P \equiv P\|!P$ $!0 \equiv 0$
Reordering of paralle processes :	Scope rules for name bindings :
$P\|Q \equiv Q\|P$ $(P\|Q)\|R \equiv P\|(Q\|R)$ $P\|0 \equiv P$	$(v\ n)0 \equiv 0$ $(v\ n)(v\ n')P \equiv (v\ n')(v\ n)P$ if $\mathrm{n} \neq \mathrm{n}'$ $(v\ n)(P\|Q) \equiv P\|(v\ n)Q$ if $n \notin \mathrm{fn}(P)$ $(v\ n)(n[P]) \equiv n[(v\ n)P]$ if$n \notin \mathrm{fn}(n)$ $(v\ n)P \equiv (v\ \mathrm{m})(P\{n/m\})$ if $\mathrm{m} \notin \mathrm{fn}(P)$

Table 2. Structural congruence: $P \equiv Q$ is the least congruence.

structure is static; (2) packets are simple data objects moving between sites, and (3) attacker processes are as expressive as sites. A formal definition of the well-formedness can be found in [19] for your reference.

Semantics. The semantics follows the approach of [7, 4] and is specified by the structural congruence relation $P \equiv Q$ in Table 2 and the reduction relation $P \rightarrow_{\mathcal{R}} Q$ in Table 3; there are two variants of reduction semantics: (1) the standard semantics ($\rightarrow$) in which $\mathcal{R}$ is universally true and thus can be ignored; (2) the *reference monitor semantics* ($\rightarrow_{\mathsf{RM}}$) that deals with annotations by taking $\mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) = (\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L})$; thus decryptions may happen only at crypt-points designated when the corresponding encryptions were made, otherwise the execution is aborted. As stated in Table 2, the structural congruence relation allows rearranging the syntactic appearance of processes; especially we enforce that $\alpha - renaming$ preserves canonicity. The movement interactions give rise to re-structuring ambients while the communication interactions do not change their hierarchy but modify the process to reflect the new binding of names. Here we adopt the standard notion $P[M/x]$ for substitution. If reference monitor semantics is concerned, the condition $\mathsf{RM}(\ell, \mathcal{L}', \ell', \mathcal{L})$ is checked by some rules. While the syntax requires us to annotate every process of local-output and input-from-child, they may be used for non-cryptographic purposes. If that is the case, we reserve a special label ϵ for those processes and adopt the set $\mathcal{D}$ to ensure annotations are trivial ones, formally

$$\langle M_1, \cdots, M_k \rangle^{\circ}_{\epsilon}[\mathrm{dest}\ \mathcal{D}]$$
$$(M'_1, \cdots, M'_j; x_{j+1}, \cdots, x_k)^{n}_{\epsilon}[\mathrm{orig}\ \mathcal{D}].P$$

Example 1. We consider the version of Wide Mouthed Frog (WMF) [6] below.

1. $A \rightarrow S : \quad A,\ [B, K]_{K_A}$
2. $S \rightarrow B : \quad [A, K]_{K_B}$
3. $A \rightarrow B : \quad [M]_K$

It establishes a secret session key K between the initiator A and the responder B, who share master keys K_A and K_B with a trusted server S respectively. Its

Movement of ambients

(In) $m\,[\,\text{in } n.\ P \,|\, Q\,] \,|\, n\,[R] \rightarrow_{\mathcal{R}} n\,[\,m\,[P \,|\, Q\,] \,|\, R]$

(Out) $n\,[\,m\,[\,\text{out } n.\ P \,|\, Q\,] \,|\, R] \rightarrow_{\mathcal{R}} m\,[P|Q] \,|\, n\,[R]$

Execution in context :

$$\frac{P \rightarrow_{\mathcal{R}} Q}{(v\ n)P \rightarrow_{\mathcal{R}} (v\ n)Q} \qquad \frac{P \rightarrow_{\mathcal{R}} Q}{(vk\ n)P \rightarrow_{\mathcal{R}} (vk\ n)Q} \qquad \frac{P \equiv P' \wedge P' \rightarrow_{\mathcal{R}} Q' \wedge Q' \equiv Q}{P \rightarrow_{\mathcal{R}} Q}$$

$$\frac{P \rightarrow_{\mathcal{R}} Q}{P \,|\, R \rightarrow_{\mathcal{R}} Q \,|\, R} \qquad \frac{P \rightarrow_{\mathcal{R}} Q}{n\,[P] \rightarrow_{\mathcal{R}} n\,[Q]}$$

Communication :

(Com 1) $$\frac{\wedge_{i=1}^{j} M_i = M_i' \ \wedge \mathcal{R}(\ell, \{\epsilon\}, \epsilon, \mathcal{L})}{\langle M_1, \cdots, M_k\rangle_{\ell}^{\circ}[\text{dest } \mathcal{L}] \mid (M_1', \cdots, M_j'; x_{j+1}, \cdots, x_k)^{\circ}.P \rightarrow_{\mathcal{R}} P\{M_{j+1}/x_{j+1}\} \cdots \{M_k/x_k\}}$$

(Com 2) $$\frac{\wedge_{i=1}^{j} M_i = M_i'}{\langle M_1, \cdots, M_k\rangle^{n} \mid n[(M_1', \cdots, M_j'; x_{j+1}, \cdots, x_k)^{\circ}.P|Q] \rightarrow_{\mathcal{R}} n[P\{M_{j+1}/x_{j+1}\} \cdots \{M_k/x_k\}|\ Q]}$$

(Com 3) $$\frac{\wedge_{i=1}^{j} M_i = M_i' \wedge \mathcal{R}(\ell, \{\epsilon\}, \epsilon, \mathcal{L})}{\langle M_1, \cdots, M_k\rangle_{\ell}^{\circ}[\text{dest } \mathcal{L}] \mid n[(M_1', \cdots, M_j'; x_{j+1}, \cdots, x_k)^{\uparrow}.P|Q] \rightarrow_{\mathcal{R}} n[P\{M_{j+1}/x_{j+1}\} \cdots \{M_k/x_k\}|\ Q]}$$

(Com 4) $$\frac{\wedge_{i=1}^{j} M_i = M_i'}{(M_1', \cdots, M_j'; x_{j+1}, \cdots, x_k)^{\circ}.P \mid n[\langle M_1, \cdots, M_k\rangle^{\uparrow}|R] \rightarrow_{\mathcal{R}} P\{M_{j+1}/x_{j+1}\} \cdots \{M_k/x_k\} \mid n[R]}$$

(Com 5) $$\frac{\wedge_{i=1}^{j} M_i = M_i' \wedge \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{(M_1', \cdots, M_j'; x_{j+1}, \cdots, x_k)_{\ell'}^{n}[\text{orig } \mathcal{L}'].P \mid n[\langle M_1, \cdots, M_k\rangle_{\ell}^{\circ}[\text{dest } \mathcal{L}]|R] \rightarrow_{\mathcal{R}} P\{M_{j+1}/x_{j+1}\} \cdots \{M_k/x_k\} \mid n[R]}$$

Table 3. Transition relation: $P \rightarrow Q$.

ABoxed Ambients specification is then given by:

$$\begin{array}{l}
(v\ K_A)(v\ K_B) \\
(\ A[(v\ K)\ K_A[\text{out } A.\ \text{in } S.\ (\langle A\rangle^{\uparrow} | \langle B,\ K\rangle_{A_1}^{\circ}[\text{dest } \mathrm{S}_1])] \mid \\
\qquad\qquad (v\ M)\ K[\text{out } A.\ \text{in} B.\ \langle M\rangle_{A_2}^{\circ}[\text{dest } \mathrm{B}_2]] \\
\mid \\
\quad S[(A;)^{\circ}.(B;\ y_K)_{S_1}^{K_A}[\text{orig } \mathrm{A}_1]. \\
\qquad\qquad K_B[\text{out } S.\ \text{in } B.\langle A,\ y_K\rangle_{S_2}^{\circ}[\text{dest } \mathrm{B}_1]] \\
\mid \\
\quad B[(A;\ z_K)_{B_1}^{K_B}[\text{orig } \mathrm{S}_2].(;z)_{B_2}^{z_K}[\text{orig } \mathrm{A}_2]])
\end{array}$$

At first A generates a new session key K by the restriction $(\nu\ K)$ and then sends S a packet named by the key K_A. After the packet K_A moves into S, the plain message A is delivered to server's mailbox while the encrypted values (B, K) can be read only by the enclosing ambient knowing the master key K_A. On the other side, the server acquires and checks initiator's name A by local-input and then decrypts the encrypted part of the message with input-from-child where the reference monitor checks the authentication intentions. If the decryption succeeds, the server continues checking whether B is the responder's name; if that is the case, it stores the session key K in the placeholder y_K. The left part of the process is encoded in the similar way as illustrated above and the explanation, therefore, is straightforward. □

3 Control Flow Analysis

The aim of our analysis is to safely estimate when RM can cease the computation of a process. To achieve this goal, we shall develop an analysis for extracting the following information:

- γ: $\mathbf{C} \to \mathcal{P}(\mathbf{C} \cup \lfloor M \rfloor)$ that for every ambient name approximates which ambients and capabilities may be contained.
- κ: $\mathbf{C} \to \mathcal{P}((\mathbf{C} \cup \lfloor M \rfloor)^*)$ that for every ambient name records the tuples of messages that may show up in an ambient's mailbox.
- ρ: $\mathbf{V} \to \mathcal{P}(\mathbf{C} \cup \lfloor M \rfloor)$ that for every variable records the tuples of possible values including names and capabilities.
- φ: $\mathcal{P}(\mathcal{D} \times \mathcal{D})$ that describes the possible violation of authenticity.

The judgement of the analysis takes the form

$$(\gamma,\kappa,\rho) \models^{\mu} P : \varphi$$

and says that when the subprocess P (of $P_\star$) is enclosed within an ambient μ then as P evolves γ will reflect the contents of the ambients , κ will contain the messages of ambients' mail boxes, ρ will approximate all the bindings of names, and φ (of the form (ℓ, ℓ')) indicates something encrypted at ℓ was unexpectedly decrypted at ℓ'. The analysis is specified in Table 4 for all non-communication primitives and in Table 5 for communication related ones.

In Table 4 the rules for *restriction, replication* and *parallel composition* ensure the analysis is valid for the immediate subprocesses while the rule for the *inactive process* enforces no restriction on the analysis result.

For an ambient process the analysis first records that the ambient n is inside the ambient $*$ and then continues analyzing the process P within the updated environment. Here the auxiliary functions $\mathcal{M}_\rho : M \to \mathcal{P}(\lfloor M \rfloor)$ and $\mathcal{N}_\rho : \mathbf{Name} \to \mathcal{P}(\mathbf{C})$ map a variable to a set of canonical capabilities and values respectively

$$\begin{array}{cc} \mathcal{N}_\rho(x) = \rho(\lfloor x \rfloor) \cap \mathbf{C} & \mathcal{N}_\rho(c) = \{\lfloor c \rfloor\} \\ \hline \mathcal{M}_\rho(\text{in } n) = \{\text{in } \mu \mid \mu \in \mathcal{N}_\rho(n)\} & \mathcal{M}_\rho(x) = \rho(\lfloor x \rfloor) \\ \mathcal{M}_\rho(\text{out } n) = \{\text{in } \mu \mid \mu \in \mathcal{N}_\rho(n)\} & \mathcal{M}_\rho(c) = \{\lfloor c \rfloor\} \end{array}$$

$(\gamma,\kappa,\rho) \models^* (v\ n)P : \varphi$	iff	$(\gamma,\kappa,\rho) \models^* P : \varphi$
$(\gamma,\kappa,\rho) \models^* (vk\ n)P : \varphi$	iff	$(\gamma,\kappa,\rho) \models^* P : \varphi$
$(\gamma,\kappa,\rho) \models^* 0 : \varphi$	iff	true
$(\gamma,\kappa,\rho) \models^* P_1 \mid P_2 : \varphi$	iff	$(\gamma,\kappa,\rho) \models^* P_1 : \varphi \wedge (\gamma,\kappa,\rho) \models^* P_2 : \varphi$
$(\gamma,\kappa,\rho) \models^* !P : \varphi$	iff	$(\gamma,\kappa,\rho) \models^* P : \varphi$
$(\gamma,\kappa,\rho) \models^* n[P] : \varphi$	iff	$\forall \mu \in \mathcal{N}_\rho(n) : \mu \in \gamma(*) \wedge (\gamma,\kappa,\rho) \models^\mu P : \varphi$
$(\gamma,\kappa,\rho) \models^* \text{in } n.P : \varphi$	iff	$\mathcal{M}_\rho(\text{in } n) \subseteq \gamma(*) \wedge (\gamma,\kappa,\rho) \models^* P : \varphi \wedge$ $\forall \text{in } \mu \in \mathcal{M}_\rho(\text{in } n) : \varphi_{\text{in}}(\mu)$
$(\gamma,\kappa,\rho) \models^* \text{out } n.P : \varphi$	iff	$\mathcal{M}_\rho(\text{out } n) \subseteq \gamma(*) \wedge (\gamma,\kappa,\rho) \models^* P : \varphi \wedge$ $\forall \text{out } \mu \in \mathcal{M}(\text{out } n) : \varphi_{\text{out}}(\mu)$
$(\gamma,\kappa,\rho) \models^* n.P : \varphi$	iff	$\mathcal{M}_\rho(n) \cap M \subseteq \gamma(*) \wedge (\gamma,\kappa,\rho) \models^* P : \varphi \wedge$ $\forall \text{in } \mu \in \mathcal{M}_\rho(n) : \varphi_{\text{in}}(\mu) \wedge$ $\forall \text{out } \mu \in \mathcal{M}_\rho(n) : \varphi_{\text{out}}(\mu)$

Table 4. Analysis specification (1): $(\gamma,\ \kappa,\ \rho) \models^* P$.

The last three clauses deal with prefixed processes. In each case all potential capabilities inside the current ambient are recorded by γ and then the continuation process is analyzed; the following closure conditions referred by the clauses serve the purpose of reflecting the semantics of in- and out- capabilities into the analysis.

$$\begin{array}{lll} \varphi_{\text{in}}(\mu) & \text{iff} & \forall \mu^a, \mu^p : \text{in } \mu \in \gamma(\mu^a) \wedge \mu^a \in \mathbf{C_p} \\ & & \wedge \mu^a \in \gamma(\mu^p) \wedge \mu \in \gamma(\mu^p) \Rightarrow \mu^a \in \gamma(\mu) \\ \varphi_{\text{out}}(\mu) & \text{iff} & \forall \mu^a, \mu^p : \text{out } \mu \in \gamma(\mu^a) \wedge \mu^a \in \mathbf{C_p} \\ & & \wedge \mu^a \in \gamma(\mu) \wedge \mu \in \gamma(\mu^g) \Rightarrow \mu^a \in \gamma(\mu^g) \end{array}$$

Now turn to the clauses in Table 5. The clause for local-output first collects the potential values $\mathcal{M}(M_i)$ of every capability M_i in a message and records all k-tuples of such messages $\langle v_1, v_2, \cdots, v_k \rangle$ into the local mailbox. Compared to local-output, the clauses for output-to-parent and out-to-child do not update local mailbox but store messages into the mailboxes of possible parents and children of the current ambient respectively.

The clause for local-input $(M_1, \cdots, M_j; x_{j+1}, \cdots, x_k)^\circ.P$ retrieves the local mailbox to look for the k-tuple messages whose first j elements are pointwise inside $\mathcal{M}_\rho(M_i)$ for $1 \leq i \leq j$. Then the new bindings of names are recorded by the analysis component ρ for variables $x_{j+1}, \cdots, x_k$ respectively. Finally $\mathsf{RM}(\ell,\mathcal{D},\epsilon,\mathcal{L})$ is checked for authentication; the special crypt-point ϵ and set $\mathcal{D}$ are inserted by the rule of local-input to check if any encrypted message may

$(\gamma,\kappa,\rho) \models^* \langle M_1,\cdots,M_k\rangle^\circ_\ell[\text{dest } \mathcal{L}] : \varphi$ iff $\forall v_1,\cdots,v_k\colon \wedge_{i=1}^k v_i \in \mathcal{M}_\rho(M_i)$
$\Rightarrow \langle v_1,\cdots,v_k\rangle_\ell[\text{dest } \mathcal{L}] \in \kappa(*)$

$(\gamma,\kappa,\rho) \models^* \langle M_1,\cdots,M_k\rangle^N : \varphi$ iff $\forall \mu \in \mathcal{N}_\rho(N) : \mu \in \gamma(*) \wedge$
$\forall v_1,\cdots,v_k\colon \wedge_{i=1}^k v_i \in \mathcal{M}_\rho(M_i)$
$\Rightarrow \langle v_1,\cdots,v_k\rangle_\epsilon[\text{dest } \mathcal{D}] \subseteq \kappa(\mu)$

$(\gamma,\kappa,\rho) \models^* \langle M_1,\cdots,M_k\rangle^\uparrow : \varphi$ iff $\forall \mu : * \in \gamma(\mu) \wedge \forall v_1,\cdots,v_k\colon \wedge_{i=1}^k v_i \in \mathcal{M}_\rho(M_i)$
$\Rightarrow \langle v_1,\cdots,v_k\rangle_\epsilon[\text{dest } \mathcal{D}] \subseteq \kappa(\mu)$

$(\gamma,\kappa,\rho) \models^* (M_1,\cdots,M_j;x_{j+1},\cdots,x_k)^\circ.P : \varphi$ iff
$\langle v_1,\cdots,v_k\rangle_\ell[\text{dest } \mathcal{L}] \in \kappa(*)\colon \wedge_{i=1}^j v_i \in \mathcal{M}_\rho(M_i)$
$\Rightarrow \wedge_{i=j+1}^k v_i \in \rho(x_i) \wedge (\neg \mathsf{RM}(\ell,\mathcal{D},\epsilon,\mathcal{L}) \Rightarrow (\ell,\epsilon) \in \varphi) \wedge (\gamma,\kappa,\rho) \models^* P : \varphi$

$(\gamma,\kappa,\rho) \models^* (M_1,\cdots,M_j;x_{j+1},\cdots,x_k)^N_{\ell'}[\text{orig } \mathcal{L}'].P : \varphi$ iff
$\forall \mu \in \mathcal{N}_\rho(N) : \mu \in \gamma(*) \wedge \forall \langle v_1,\cdots,v_k\rangle_\ell[\text{dest } \mathcal{L}] \in \kappa(\mu) : \wedge_{i=1}^j v_i \in \mathcal{M}_\rho(M_i)$
$\Rightarrow \wedge_{i=j+1}^k v_i \in \rho(x_i) \wedge (\neg \mathsf{RM}(\ell,\mathcal{L}',\ell',\mathcal{L}) \Rightarrow (\ell,\ell') \in \varphi) \wedge (\gamma,\kappa,\rho) \models^* P : \varphi$

$(\gamma,\kappa,\rho) \models^* (M_1,\cdots,M_j;x_{j+1},\cdots,x_k)^\uparrow.P : \varphi$ iff
$\forall \mu : * \in \gamma(\mu) \wedge \forall \langle v_1,\cdots,v_k\rangle_\ell[\text{dest } \mathcal{L}] \in \kappa(\mu)) : \wedge_{i=1}^j v_i \in \mathcal{M}_\rho(M_i)$
$\Rightarrow v_{j+1} \in \rho(x_j) \wedge \cdots \wedge v_k \in \rho(x_k) \wedge (\neg \mathsf{RM}(\ell,\mathcal{D},\epsilon,\mathcal{L}) \Rightarrow (\ell,\epsilon) \in \varphi) \wedge (\gamma,\kappa,\rho) \models^* P : \varphi$

Table 5. Analysis specification (2): $(\gamma,\ \kappa,\ \rho) \models^* P$.

be read unexpectedly. For the rule of input-from-parent and input-from-child we retrieve the mailboxes of possible parents and children of the current ambient respectively. The left part of the rule is quite similar to that of local-input except that no annotations are implicitly added in the rule of input-from-child as they have been declared explicitly. Especially we do not need a rule for the attacker process as it could be any processes (well-formed) whose analysis has been declared as above.

Semantic Properties. We prove the correctness of the analysis w.r.t. the operational semantics of ABoxed Ambients. It is convenient to prove the following lemmata. The first says that estimates keep valid for substitution of closed terms for variables. The second states that an estimate valid for a process P is also valid for every process congruent to P.

Lemma 1. $(\gamma,\kappa,\rho) \models^\mu P : \varphi$ *and* $\lfloor M \rfloor \in \rho(\lfloor x \rfloor)$ *imply* $(\gamma,\kappa,\rho) \models^\mu P\{M/x\} : \varphi$.

Lemma 2. *If* $P \equiv Q$ *then* $(\gamma,\kappa,\rho) \models^\mu P : \varphi$ *iff* $(\gamma,\kappa,\rho) \models^\mu Q : \varphi$.

We are now ready to state the subject reduction result, which says our analysis is semantically correct for both two variants of semantics:

Theorem 1. *If* $P \rightarrow_{\mathcal{R}} Q$ *and* $(\gamma,\kappa,\rho) \models^{\mu} P : \varphi$ *then* $(\gamma,\kappa,\rho) \models^{\mu} Q : \varphi$.

Finally we conclude that the analysis can correctly predict when we can safely remove the reference monitor:

Theorem 2. *If* $(\gamma,\kappa,\rho) \models^{\mu} P : \emptyset$ *then* RM *can not abort* P.

Example 2. For the ABoxed Ambients specification of WMF specified in Example 1, an estimate satisfying $(\gamma, \kappa, \rho) \models^{\star}$ WMF $: \varphi$ is given by

$$
\begin{array}{lllllll}
\gamma: & n_\star & \mapsto & \{A,S,B,K_A,K_B,K\} & A & \mapsto & \{K_A,K\} \\
 & S & \mapsto & \{K_A,K_B\} & B & \mapsto & \{K_B,K\} \\
 & K_A & \mapsto & \{\text{out } A, \text{in } S\} & K & \mapsto & \{\text{out } A, \text{in } B\} \\
 & K_B & \mapsto & \{\text{out } S, \text{in } B\} & & & \\
\kappa: & A & \mapsto & \{\langle A\rangle_\epsilon[\text{dest } \mathcal{D}]\} & B & \mapsto & \emptyset \\
 & S & \mapsto & \{\langle A\rangle_\epsilon[\text{dest } \mathcal{D}]\} & K_A & \mapsto & \{\langle B,K\rangle_{A_1}[\text{dest } S_1]\} \\
 & K & \mapsto & \{\langle M\rangle_{A_2}[\text{dest } B_2]\} & K_B & \mapsto & \{\langle A,K\rangle_{S_2}[\text{dest } B_1]\} \\
\rho: & y_K & \mapsto & \{K\} & z_K & \mapsto & \{K\} \\
 & z & \mapsto & \{M\} & & &
\end{array}
$$

and $\varphi = \emptyset$ predicting that RM can not abort the process computation. □

4 Modelling Network Attacker

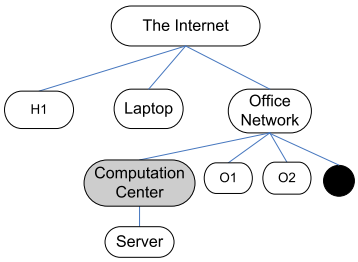
Protocols are executed in a multi-location environment where there may be malicious attackers in some of places. In a flat space of network, we usually take the form $n_\star[P \mid \bullet]$ in which P and $\bullet$ represent the implementation of a system and its working environment respectively. For the hierarchical network, however, there may be several local networks accessible to the attacker. Thus we must provide our assumption about which local networks the attacker may reside in. Suppose the attacker is on the network represented by the distinguished ambient $n_\star$ or a site ambient a, we declare attacker processes as one of top level processes of them, formally $n_\star[P \mid \bullet]$ or $a[Q \mid \bullet]$. Below we shall call a process without attackers inside *target process*. We can use $P_{sys}[0/\bullet]$ to get the target process from a system implementation P_{sys}.

To characterize all capabilities of network attackers, we aim at finding a parameterized formula $\mathcal{F}^{\text{A\_DY}}_{\mathsf{RM}}(*)$; whenever an estimate $(\gamma, \kappa, \rho, \varphi)$ satifies $\mathcal{F}^{\text{A\_DY}}_{\mathsf{RM}}(*)$ then $(\gamma, \kappa, \rho) \models^{*} R : \varphi$ for all attackers R. Before we proceed to define such a formula, we must declare attackers' power on the network at first. The pioneering research in [9] describes the attacker capabilities as four conditions: (1) receiving messages by eavesdropping, (2) decrypting messages using the key they know, (3) constructing new messages (encrypted or plain), and (4) sending messages they have. Here the conditions (1) and (4) are not clear enough if the principal of local networks are concerned. For the first condition we need to provide assumption that which location(s) the attacker can overhear; for the fourth one we should clarify that which location(s) the attacker can send message to. We

turn to the following adjusted Dolev-Yao condition; the design idea is that to guarantee any flaw of a protocol can be detected, the attacker should be able to control over any network resource he might gain in the real world.

a. Eavesdropping on any messages presenting in the attacker-nested location (declared by the attacker process);
b. Decrypting messages using the key the attacker knows;
c. Constructing both encrypted and plain messages;
d. Sending messages to any attacker-reachable sites;
e. Initially the attacker has some knowledge and a private channel is allocated for all attackers to share information with each other.

While the first three conditions are straightforward, we explain the last two in detail. The forth item declares that the attacker can deliver messages to any reachable site. We define the concept "reachable" based on the knowledge of the attacker: there is a route (consists of a series of sites) from attacker-nested place to a destination along which each name of the site is known by the attacker. For example, consider the network in Figure 1 again and suppose the attacker resides in the office. We then colored the tree in Figure 1 as below.



where grey nodes denotes attacker-invisible sites and white nodes represent the sites whose names are knowable to the attacker. As the figure shows, Computation Center is not reachable to the attacker-composed messages. Neither is Server although its name is known by the bad guy. Finally the fifth item allows attackers attack system by collusion. This is a strong assumption about attacker's capability: it always takes time to broadcast messages among attackers in reality; also we maximize attacker's power by assuming all malicious entities share information with each other. However, this only implies that we may get over-estimates but no flaw of a security protocol can be left over.

We follow the approach of [2] and state that a target process P is of type $(\mathcal{N}_{\mathrm{f}}, \mathcal{A}_{\mathcal{K}})$ whenever: (1) P is closed, (2) its free names are in $\mathcal{N}_{\mathrm{f}}$, and (3) all the arities used by input and output are in $\mathcal{A}_{\mathcal{K}}$. We can easily find minimal $\mathcal{N}_{\mathrm{f}}$ and $\mathcal{A}_{\mathcal{K}}$ so that P is of type $(\mathcal{N}_{\mathrm{f}}, \mathcal{A}_{\mathcal{K}})$. To charatacterise all attackers R, we have adopted a few assumptions and applied techniques to translate R into its semantically equivalent process $\overline{R}$ in order to have control over the infinite names and labels that attackers may have. Accordingly we have specified the formula $\mathcal{F}^{\mathrm{A\_DY}}_{\mathsf{RM}}(*)$; the idea is to add a series of constraints to an estimate so that the

adjusted conditions can be implied from the estimate. For detailed description, please refer to [19].

We can establish the correctness of the adjusted Dolev-Yao condition for ABoxed Ambients in the following two theorems. The first state that estimates satisfying $\mathcal{F}_{\mathsf{RM}}^{\mathrm{A\_DY}}(*)$ are also valid for all attackers in site $*$.

Theorem 3. *If an estimate* $(\gamma,\kappa,\rho,\varphi)$ *satisfies* $\mathcal{F}_{\mathsf{RM}}^{\mathrm{A\_DY}}(*)$ *of type* $(\mathcal{N}_{\mathrm{f}},\mathcal{A}_{\mathcal{K}})$ *then* $(\gamma,\kappa,\rho) \models^{*} \overline{R}$*:* φ *for all well-formed processes* R *of type* $(\mathcal{N}_{\mathrm{f}},\mathcal{A}_{\mathcal{K}})$*.*

Now assume $n_{\star}[P_{sys}]$ is the implementation of a system and a set of attacker-nesting places of P_{sys} is in the set I, we prove that estimates satisfying $\wedge_{*\in I}\mathcal{F}_{\mathsf{RM}}^{\mathrm{A\_DY}}(*)$ are valid for all attackers in the system:

Theorem 4. *If* $(\gamma,\kappa,\rho) \models^{\star} P_{sys}[0/\bullet] : \varphi$ *and* $(\gamma,\kappa,\rho,\varphi)$ *satisfies* $\wedge_{*\in I}\mathcal{F}_{\mathsf{RM}}^{\mathrm{A\_DY}}(*)$ *of type* $(\mathcal{N}_{\mathrm{f}},\mathcal{A}_{\mathcal{K}})$*, then* $(\gamma,\kappa,\rho) \models^{\star} P_{sys}[\overline{R}/\bullet] : \varphi$ *for all attackers* R*.*

5 ABox-Ambients Tool

We aim at developing an automatic tool to compute our control flow analysis correctly and efficiently. It can be shown that there always is a least estimate of γ, κ, ρ *and* φ for any process P such that $(\gamma,\kappa,\rho) \models^{\star} P : \varphi$. The aim of the tool is to compute such a least $(\gamma, \kappa, \rho, \varphi)$ for a given process. The generic strategy of implementing constraint-based analysis is to translate an analysis into a suitable constraint language and then compute the least estimate of these constraints with a standard constraint solver. We adopt Succinct Solver 2.0 [16], an expressive fragment of first-order predicate logic, as our constraint solver to obtain an efficient tool. The solver takes constraints encoded with Alternation-free Least Fixed Point logic (ALFP) as input and gives the least solution of a program analysis as output. The transforming of the analysis into ALFP proceeds in three steps. First we transform the analysis from succinct form into its verbose form [17] so that every analysis component has global scope. This is because the ALFP recognized by Succinct Solver can not provide scoping mechanisms for predicates. Second we translate the analysis and the attacker formulae into ALFP. This is conducted in a series of straightforward encodings, for instance, representing sets as predicates, and encoding annotations in communication primitives. Finally the analysis and the attacker formulae are turned into a generation function $\mathcal{G}$ that takes a process as argument and returns its analysis in the form of ALFP formulae.

As explained in [16], the time for solving a formula in Succinct Solver is polynomial in the size of a finite universe of atomic values, e.g. canonical names and capabilities, over which a formula is interpreted. Suppose the size of the universe is N, then a simple worst-case estimate of execution time is about $\mathcal{O}(N^{1+\tau})$ where τ is the maximal nesting depth of quantifiers in the clause. For our implementation, the depth of nesting is mainly given by the length of the sequences specified in communication.

6 Protocol Validation

Protocol validation is usually based on many assumptions. For instance, most formal techniques assume that cryptography is perfect, the master keys are always securely stored and retrieved. In this section we first discuss how to use our calculus to model key-store and key-retrieving and thereby protocols can be validated under fewer assumptions. By doing so, we expect that the approach can provide system designer more useful information. We then validate WMF and its two variants in a series of configurations, a set of assumptions about the network hierarchy, the locations of different roles and attackers. In all the experiments we have taken the number of each role (except server) to be 3 in order to ensure that the man-in-the-middle attack can be modeled.

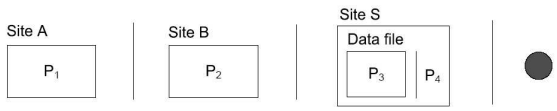
Validating Protocol with Key-retrieving. Our first attempt is to model a data file storing master keys on the server in plain text. This can be formalised as:

$$\text{KeyTable} \;=\; datafile[!\langle n_1, K_1\rangle^{\circ}_{\epsilon}[\text{dest } \mathcal{D}]] | \cdots | !\langle n_m, K_{\text{m}}\rangle^{\circ}_{\epsilon}[\text{dest } \mathcal{D}]]$$

where n_i and K_i are the identity of a principle and its key respectively. Replication '!' is used to present the data of the table is persistent. Querying the table can be encoded as:

$$\text{Keytable}|(n_i; y_{\text{k}})^{\text{datafile}}_{\epsilon}[\text{dest}\mathcal{D}]. \cdots .y_{\text{k}} \cdots$$

where we take advantage of pattern match to check the name n_i in input and acquire its key by variable binding. Following this design idea, we can update the specification of Example 1 and validate WMF under the configuration whose ambient representation is visualized as:



The experiment result shows no flaw is found in the system. Next suppose there are a large amount of secret keys to store and then a dedicated database server is assigned to support the service of an authentication server. In the real life the two servers are usually located in a secure area, e.g. a local network, to which no attackers can physically access. We then validate WMF on the network whose ambient structure is presented as below.

Here the database query is described by narration

$$\begin{array}{lllll} 1. & S & \rightarrow & DBS: & A \\ 2. & DBS & \rightarrow & S: & A, K_A \end{array}$$

Our experiment result shows that the protocol may be flawed and φ is

$$\{(A_{2i}, \ell_\bullet) | 1 \leq i \leq n\} \cup \{(S_2, \ell_\bullet)\} \cup \{(\ell_\bullet, S_1)\}$$
$$\cup \{(\ell_\bullet, B_{1j}) | 1 \leq j \leq n\} \cup \{(\ell_\bullet, B_{2j}) | 1 \leq j \leq n\}$$

Actually the protocol is flawed as illustrated by below two attacks.

(i)
$$\begin{array}{llll} M_A & \rightarrow S: & A, [B, K_{M_1}]_{K_{M_2}} \\ M_{DB} & \rightarrow S: & A, K_{M_2} \\ S & \rightarrow B: & [A, K_{M_1}]_{K_B} \\ M_A & \rightarrow B: & [m]_{K_{M_1}} \end{array}$$

(ii)
$$\begin{array}{llll} A & \rightarrow S & : & A, [B, K]_{K_A} \\ M_{DB} & \rightarrow S & : & B, K_M \\ S & \rightarrow M_B & : & [A, K]_{K_M} \\ A & \rightarrow B & : & [m]_K \end{array}$$

For attack (i), B finally believes that he is getting message from A but he is actually reading messages composed by the attacker. For attack (ii) the attacker cheats the server S by sending it a fake master key K_M and finally the attacker can decrypt any message sent from A to B. The root cause of the flaw is that the authentication server can not distinguish the packets from the database server with those from attackers. We can fix the problem by either encrypting messages sent between the servers or simply modifying their communication as:

$$\begin{array}{llll} 1. & S & \rightarrow DBS: & A \\ 2. & DBS & \rightarrow S: & u, A, K_A \end{array}$$

where a new name u is introduced and initially known only by the two servers. Our experiment shows that the protocol is flawless for both the two solutions.

Optimizing Protocol in Hierarchical Networks. We now consider two variants of WMF: one where the first message ($A \rightarrow S$) is not encrypted and one where the second message ($S \rightarrow B$) is not encrypted; the protocol narration is as below.

Variant 1:
$$\begin{array}{lll} A \rightarrow S: & u_1, A, B, K \\ S \rightarrow B: & [A, K]_{K_B} \\ A \rightarrow B: & [m]_K \end{array}$$

Variant 2:
$$\begin{array}{lll} A \rightarrow S: & A, [B, K]_{K_A} \\ S \rightarrow B: & u_2, A, K \\ A \rightarrow B: & [m]_K \end{array}$$

Here we assume u_1 is initially only known by A and S while u_2 is restricted over B and S. We validate the two protocols in a number of configurations; the experiment results are summarized in Table 6.

As shown in the first line of the table, the analysis reports that both the two variants are flawed since the session key K can be acquired by the attacker. For the second configuration, we assume the initiators and the server are located in the office that is not accessible to the attacker. Now the validation results show the first variant is still flawed but this time the second is secure. This is because the attacker can not overhear or intercept messages on the office network and that actually provides a private channel for the initiators and the server. Now we state Variant 2 has advantages over WMF in efficiency and space-consumption considering both of them are secure because (1) the variant saves time in encrypting and decrypting values that is usually the most time-consuming operations in security protocol, and (2) it sharply reduces the size of a data file or data base by storing much less master keys than before. Similarly we

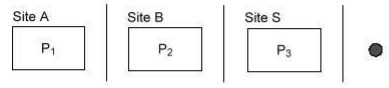
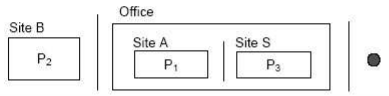
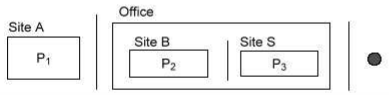
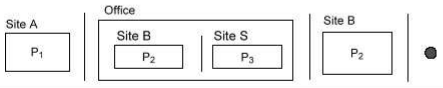
Configuration	WMF-Variant 1	WMF-Variant 2
	$\{(A_{1i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{2_j}) \mid 1 \le j \le n\}$	$\{(A_{2i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{1_j}) \mid 1 \le j \le n\}$
	$\{(A_{1i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{2_j}) \mid 1 \le j \le n\}$	$\varnothing$
	$\varnothing$	$\{(A_{2i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{1_j}) \mid 1 \le j \le n\}$
	$\{(A_{2i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{1_j}) \mid 1 \le j \le n\}$	$\{(A_{2i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{1_j}) \mid 1 \le j \le n\}$
	$\{(A_{2i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{1_j}) \mid 1 \le j \le n\}$	$\{(A_{2i}, \ell_\bullet) \mid 1 \le i \le n\} \cup$ $\{(\ell_\bullet, B_{1_j}) \mid 1 \le j \le n\}$

Table 6. Experiments on validating protocols in hierarchical networks.

switch the position of the initiators for the responders in the third configuration; this time the first variant is secure as expected (see third row of Table 6).

The fourth configuration assumes the responders may appear on both the Internet and the office while the last one supposes a malicious guy gain access to the office. In both the two cases the variants are flawed as the attacker can acquire the session key K and thus the security of the protocols is compromised.

Summarizing the results of the experiment, we conclude that it is possible to optimize a protocol by considering network structures and principals' locations; in particular, the analysis can help system designers check whether the adapted protocol still guarantee authentication and provide information to track flaws if there are any.

7 Conclusion

We have shown that hierarchical networks and protocols applied on such networks may be formalized as ABoxed Ambients processes so that a static analysis can pinpoint a wide-variety of errors in security protocols. We have also presented a new attacker model based on the Dolev-Yao model in order to comply with the special network considered in this paper. We have argued that the model gives the attacker reasonable abilities to conduct passive and positive attack to protocols.

The analysis has been implemented using the Succinct Solver 2.0 and has then been applied to a number of examples. We would like to extend our cal-

culus to deal with asymmetric cryptography. Also it would be interesting to see how the approach scales to a large protocol which is developed for the environment of hierarchial networks.

Comparison with related work. A number of formal methods have been developed in the field of protocol analysis. We shall compare them with our work in two aspects: the approaches of protocol formalism and the analysis techniques used to validate protocols. Many papers have considered to formulate protocols with process calculi such as CSP [12], CCS [10], Lysa [2, 3] and ambient calculus [18]. We consider ambient calculi as a proper choice with regard to the network of interest; the scope of the message of local communication is clearly given by the boundary of ambients. With CSP, CCS and Lysa, one may use private channels to model local communication between principals. But the resulting specification would be harder to understand compared to the original topology of the modelled network. Ambients, however, can formulate the principle of local networks in a quite nature way.

Boxed Ambients is first used to model security protocol in [18] where a control flow analysis is also developed to track communication happening on different locations. But there is no attacker defined to model the realistic environment. We also have modified the calculus for the purpose of protocol validation specially, e.g. extending the input with a pattern match to model value-checking, adding annotations to declare protocol intentions explicitly.

Based on formal protocol specification, a lot of techniques have been developed to analyze protocols automatically. Two of main trends close to our approach are type systems and model checking. Type systems have been developed for security protocol analysis, e.g. by Abadi [1] and by Gordon and Jeffery [11]. The results show that type checking in these systems can be done in polynomial time while type inference takes exponential time. In comparison, the control flow analysis presented here retains polynomial time.

Model checking is a method that explores each state in a protocol; see e.g. FDR [13], Interrogator [14] and Brutus [8]. Since the state space for security protocol is usually infinite, the approach based on state space exploration can not guarantee termination while our approach adopts approximation to deal with arbitrarily long execution sequences. On the other hand, model checking techniques are often quite efficient in finding flaws if there is any in a protocol. Thus it can be seen as complementary to control flow analysis techniques.

The major advantages of static analysis approach taken here can be summarized as: first, the least solution always exists and can be computed in low polynomial time; second, the approach is operational oriented so that the correctness of the analysis can be established w.r.t. a formal operation semantics; last but not least, the approach can be fully automated.

References

1. M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.
2. C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. R. Nielson. Static validation of security protocols. *Journal of Computer Security*, 13(3):347–390, 2005.
3. M. Buchholtz, H. R. Nielson, and F. Nielson. A calculus for control flow analysis of security protocols. *Int. J. Inf. Sec.*, 2(3-4):145–167, 2004.
4. M. Bugliesi, G. Castagna, and S. Crafa. Boxed Ambients. In *TACS*, pages 38–63, 2001.
5. M. Bugliesi, G. Castagna, and S. Crafa. Reasoning about security in mobile ambients. In *CONCUR*, pages 102–120, 2001.
6. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. In *SOSP*, pages 1–13, 1989.
7. L. Cardelli and A. D. Gordon. Mobile ambients. *Theor. Comput. Sci.*, 240(1):177–213, 2000.
8. E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with Brutus. *ACM Transactions on Software Engineering and Methodology*, 9(4):443–487, 2000.
9. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
10. R. Focardi and R. Gorrieri. A taxonomy of security properties for process algebras. *Journal of Computer Security*, 3(1):5–34, 1995.
11. A. D. Gordon and A. Jeffrey. Authenticity by typing for security protocols. *Journal of Computer Security*, 11(4):451–520, 2003.
12. G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.
13. G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR. In *TACAS*, pages 147–166, 1996.
14. J. K. Millen. The interrogator: A tool for cryptographic protocol security. In *IEEE Symposium on Security and Privacy*, pages 134–141, 1984.
15. F. Nielson, H. R. Nielson, and R. R. Hansen. Validating firewalls using flow logics. *Theor. Comput. Sci.*, 283(2):381–418, 2002.
16. F. Nielson, H. Seidl, and H. R. Nielson. A succinct solver for ALFP. *Nord. J. Comput.*, 9(4):335–372, 2002.
17. H. R. Nielson and F. Nielson. Flow Logic: A multi-paradigmatic approach to static analysis. In *The Essence of Computation*, pages 223–244, 2002.
18. H. R. Nielson, F. Nielson, and M. Buchholtz. Security for Mobility. In *FOSAD*, pages 207–265, 2002.
19. Y. Zhang. Static analysis for protocol validation in hierarchical networks. Master's thesis, Technical University of Denmark, 2005.