Static Security Analysis of Service-Oriented Systems

Flemming Nielson Hanne Riis Nielson Han Gao

Technical University of Denmark

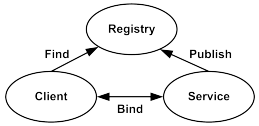
Tutorial Series on Global Computing
Lisbon, June 9th, 2009

RoadMap

- Service Oriented Systems
- Process Calculi Models
- Abstract Level
- Protocol Stack Plug-in Level
- Concrete Level
- Formalisation of Intensions by Annotations
- LySa Calculus
- Static Analysis
- Current Status: Extensions and Integration
- Conclusion

Service Oriented Systems

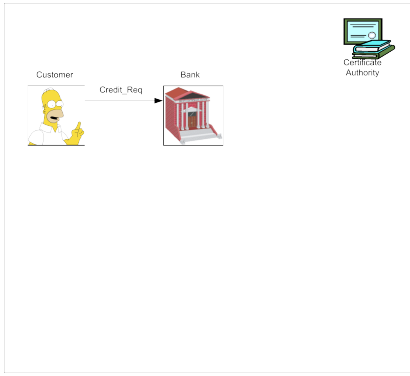
A new computing paradigm that utilises services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments.



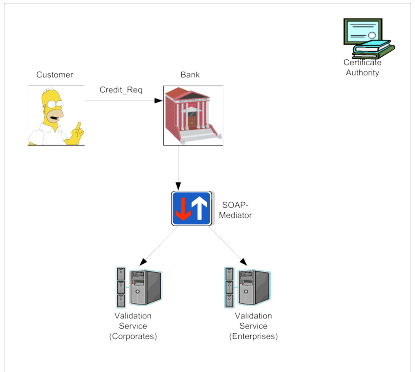
- Greater interoperability
- Loosely coupled
- Easier to integrate
- Increased reuse
- Reduce costs

SENSORIA: An EU project aiming to develop a novel comprehensive approach to the engineering of software systems for service-oriented architectures.

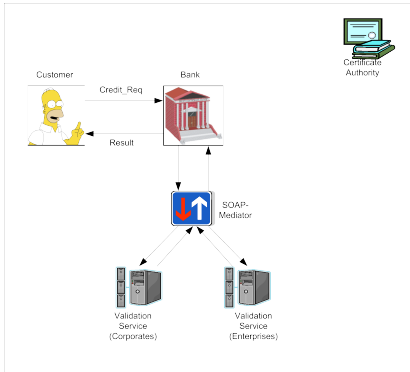
An Example of Service Oriented Applications



An Example of Service Oriented Applications



An Example of Service Oriented Applications



The Credit Request Case Study

(TLS)

1. $C \rightarrow VS : N_C$
2. $VS \rightarrow C : N_{VS}, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)$
3. $C \rightarrow VS : \mathbf{P}_{K_{VS}^+}(N_{PMS})$

(BALANCE VALIDATION REQUEST)

4. $C \rightarrow VS : \mathbf{E}_{\mathbf{H}(N_C, N_S, N_{PMS})}(Bta)$

(SERVICE INVOCATION AND ROUTING)

5. $VS \rightarrow SM : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$
6. $SM \rightarrow Ser_E : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$

(EVALUATION AND REPLY)

7. $Ser_E \rightarrow SM : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
8. $SM \rightarrow VS : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
9. $VS \rightarrow C : \mathbf{E}_{\mathbf{H}(N_C, N_{VS}, N_{PMS})}(Bta, Res)$

The Credit Request Case Study

(TLS)

1. $C \rightarrow VS : N_C$
2. $VS \rightarrow C : N_{VS}, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)$
3. $C \rightarrow VS : \mathbf{P}_{K_{VS}^+}(N_{PMS})$

(BALANCE VALIDATION REQUEST)

4. $C \rightarrow VS : \mathbf{E}_{\mathbf{H}(N_C, N_S, N_{PMS})}(Bta)$

(SERVICE INVOCATION AND ROUTING)

5. $VS \rightarrow SM : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$
6. $SM \rightarrow Ser_E : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$

(EVALUATION AND REPLY)

7. $Ser_E \rightarrow SM : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
8. $SM \rightarrow VS : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
9. $VS \rightarrow C : \mathbf{E}_{\mathbf{H}(N_C, N_{VS}, N_{PMS})}(Bta, Res)$

The Credit Request Case Study

(TLS)

1. $C \rightarrow VS : N_C$
2. $VS \rightarrow C : N_{VS}, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)$
3. $C \rightarrow VS : \mathbf{P}_{K_{VS}^+}(N_{PMS})$

(BALANCE VALIDATION REQUEST)

4. $C \rightarrow VS : \mathbf{E}_{\mathbf{H}(N_C, N_S, N_{PMS})}(Bta)$

(SERVICE INVOCATION AND ROUTING)

5. $VS \rightarrow SM : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$
6. $SM \rightarrow Ser_E : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$

(EVALUATION AND REPLY)

7. $Ser_E \rightarrow SM : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
8. $SM \rightarrow VS : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
9. $VS \rightarrow C : \mathbf{E}_{\mathbf{H}(N_C, N_{VS}, N_{PMS})}(Bta, Res)$

The Credit Request Case Study

(TLS)

1. $C \rightarrow VS : N_C$
2. $VS \rightarrow C : N_{VS}, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)$
3. $C \rightarrow VS : \mathbf{P}_{K_{VS}^+}(N_{PMS})$

(BALANCE VALIDATION REQUEST)

4. $C \rightarrow VS : \mathbf{E}_{\mathbf{H}(N_C, N_S, N_{PMS})}(Bta)$

(SERVICE INVOCATION AND ROUTING)

5. $VS \rightarrow SM : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$
6. $SM \rightarrow Ser_E : VS, Ser_E, \mathbf{S}_{K_{VS}^-}(\mathbf{P}_{K_{Ser_E}^+}(SN, Bta, \mathbf{S}_{K_{CA}^-}(VS, K_{VS}^+)))$

(EVALUATION AND REPLY)

7. $Ser_E \rightarrow SM : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
8. $SM \rightarrow VS : Ser_E, VS, \mathbf{S}_{K_{Ser_E}^-}(\mathbf{P}_{K_{VS}^+}(SN, Res))$
9. $VS \rightarrow C : \mathbf{E}_{\mathbf{H}(N_C, N_{VS}, N_{PMS})}(Bta, Res)$

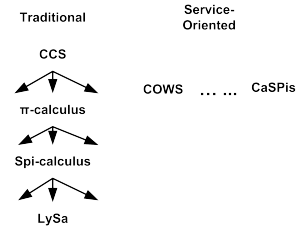
Modelling Behaviour of Systems

Systems can be described at many levels:

- Software Systems
  - Programming Languages (VHDL/Java/F#)
- Software Designs
  - Object Oriented Models (UML)
- Software Processes
  - Process Calculi (CCS,π,spi, LySa,...,COWS,CaSPis,...)

Process Calculi

Process Calculi: Tiny but powerful languages for modelling communicating systems.



- Support massive parallelism.
- Incorporate communication.
- Can be extended to handle cryptographic primitives.
- Have a formal semantics.
- Are subject to automatic analysis.
- Can be extended to model Service Oriented applications.

Traditional Process Calculi

- CCS: No values are communicated – we only have synchronisation.
- π: Only names can be communicated. Communications are through channels.
- spi: Encrypted values can be communicated. Communications are through channels.
- LySa: Encrypted values can be communicated – but we only have one (global) channel, the ether.

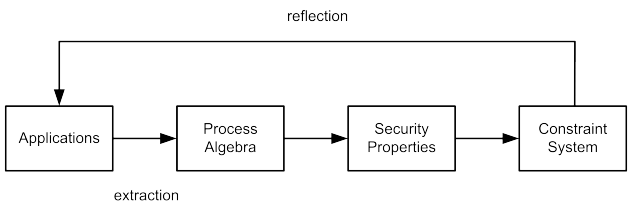
Bridging the Gap

We model Service Oriented systems at different levels

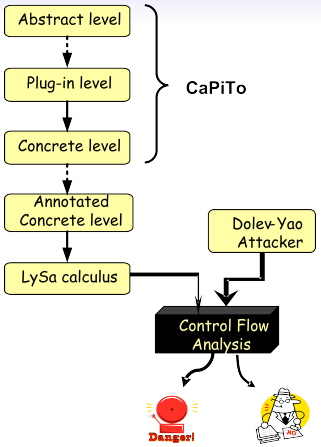
- Abstract level (as often found in academia)
- Protocol stack plug-in level (bridge the cap between the two levels)
- Concrete level (as often found in industry)

We then transform the model into LySa calculus and use the static analyser LySa tool to analyse the specification.

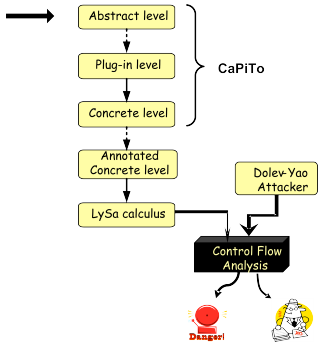
The Vision



The Framework



Next: Introduction to Abstract Level



Abstract Level Syntax (1)

value	v, w	$::=$	n	name
		$\mid$	$f(\vec{v})$	function
expr	e	$::=$	x	variable
		$\mid$	n	name
		$\mid$	$f(\vec{e})$	function
pattern	p	$::=$	$?x$	defining occurrence of variable
		$\mid$	x	applied occurrence of variable
		$\mid$	n	name

- Abstract away from both cryptography and industrial communication protocols
- Concentrate on the interaction of the services themselves.
- Distinguish between defining occurrences and applied occurrences of variables.

Abstract Level Syntax (2)

Process	P	$::=$	$\langle\vec{e}\rangle.P$	output
		$\mid$	$(\vec{p}).P$	input
		$\mid$	$(\nu\ n)P$	restriction
		$\mid$	$P_1 \mid P_2$	parallel
		$\mid$	$!P$	replication
		$\mid$	$P_1 + P_2$	nondeterministic choice
		$\mid$	0	nil
		$\mid$	$\overline{n}[\,] \cdot P$	service invocation
		$\mid$	$n[\,] \cdot P$	service provider
		$\mid$	$\uparrow \langle\vec{e}\rangle.P$	return

- $n[\,] \cdot P$ provides a service n that is to be invoked by $\overline{n}[\,] \cdot P$
- $\uparrow \langle\vec{e}\rangle.P$ returns $\vec{e}$ to the outside environment

Abstract Level Modelling of the Case Study

$$System \triangleq Client \mid Bank \mid Ser_E$$

$$Client \triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,] \cdot \Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big)$$

$$Bank \triangleq !\ credit\_req[\,] \cdot \Big((?y_{bta}).\ \overline{validate}[\,] \cdot \langle y_{bta}\rangle.(?y_{res}).\uparrow\langle y_{bta}, y_{res}\rangle.0\Big)$$

$$Ser_E \triangleq !\ validate[\,] \cdot \Big((?z_{bta}).\langle isValid(z_{bta})\rangle.0\Big)$$

Bta : Balance Total Assets

Abstract Level Modelling of the Case Study

$$
\begin{aligned}
System &\triangleq Client \mid Bank \mid Ser_E \\
Client &\triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big) \\
Bank &\triangleq !\ credit\_req[\,]\cdot\Big((?y_{bta}).\ \overline{validate}[\,]\cdot\langle y_{bta}\rangle.(?y_{res}).\uparrow\langle y_{bta}, y_{res}\rangle.0\Big) \\
Ser_E &\triangleq !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(z_{bta})\rangle.0\Big)
\end{aligned}
$$

Bta : Balance Total Assets

Abstract Level Modelling of the Case Study

$$
\begin{aligned}
System &\triangleq Client \mid Bank \mid Ser_E \\
Client &\triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big) \\
Bank &\triangleq !\ credit\_req[\,]\cdot\Big((?y_{bta}).\ \overline{validate}[\,]\cdot\langle Bta\rangle.(?y_{res}).\uparrow\langle Bta, y_{res}\rangle.0\Big) \\
Ser_E &\triangleq !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(z_{bta})\rangle.0\Big)
\end{aligned}
$$

Bta : Balance Total Assets

Abstract Level Modelling of the Case Study

$$
\begin{aligned}
System &\triangleq Client \mid Bank \mid Ser_E \\
Client &\triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big) \\
Bank &\triangleq !\ credit\_req[\,]\cdot\Big((?y_{bta}).\ \overline{validate}[\,]\cdot\langle Bta\rangle.(?y_{res}).\uparrow\langle Bta, y_{res}\rangle.0\Big) \\
Ser_E &\triangleq !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(Bta)\rangle.0\Big)
\end{aligned}
$$

Bta : Balance Total Assets

Abstract Level Modelling of the Case Study

$$
\begin{aligned}
System &\triangleq Client \mid Bank \mid Ser_E \\
Client &\triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big) \\
Bank &\triangleq !\ credit\_req[\,]\cdot\Big((?y_{bta}).\ \overline{validate}[\,]\cdot\langle Bta\rangle.(?y_{res}).\uparrow\langle Bta, isValid(Bta)\rangle.0\Big) \\
Ser_E &\triangleq !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(Bta)\rangle.0\Big)
\end{aligned}
$$

Bta : Balance Total Assets

Abstract Level Modelling of the Case Study

$$System \triangleq Client \mid Bank \mid Ser_E$$

$$Client \triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big)$$

$$Bank \triangleq !\ credit\_req[\,]\cdot\Big((?y_{bta}).\ \overline{validate}[\,]\cdot\langle Bta\rangle.(?y_{res}).\uparrow\langle Bta, isValid(Bta)\rangle.0\Big)$$

$$Ser_E \triangleq !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(Bta)\rangle.0\Big)$$

Bta : Balance Total Assets

Abstract Level Modelling of the Case Study

$$System \triangleq Client \mid Bank \mid Ser_E$$

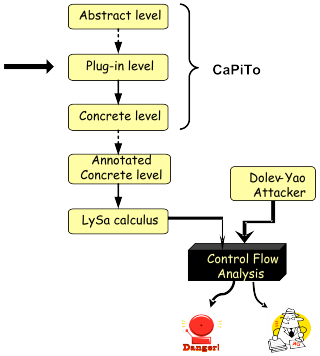
$$Client \triangleq !\,(\nu\ Bta)\ \overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle isValid\,Bta\rangle.0\Big)$$

$$Bank \triangleq !\ credit\_req[\,]\cdot\Big((?y_{bta}).\ \overline{validate}[\,]\cdot\langle Bta\rangle.(?y_{res}).\uparrow\langle Bta, isValid(Bta)\rangle.0\Big)$$

$$Ser_E \triangleq !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(Bta)\rangle.0\Big)$$

Bta : Balance Total Assets

Next: Introduction to Protocol Stack Plug-in Level



Protocol Stack Plug-in (general) Level Syntax

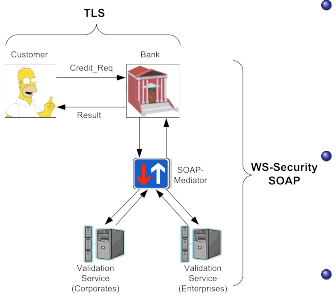
We wish to be precise about which protocols we intend to use.

$$\begin{array}{lcl}
v, w & ::= & n \mid f(\vec{v}) \\
e & ::= & x \mid n \mid f(\vec{e}) \\
p & ::= & ?x \mid x \mid n \\
P & ::= & \langle\vec{e}\rangle.P \mid (\vec{p}).P \mid \\
 & & (\nu\ n)P \mid P_1|P_2 \mid !P \mid P_1 + P_2 \mid 0 \mid \\
 & & \overline{n}[ps].P \mid n[ps].P \mid \uparrow\langle\vec{e}\rangle.P \\
ps & ::= & pi \mid pi; ps \\
pi & ::= & name, param_1, \cdots, param_k
\end{array}$$

- *ps* is a protocol stack containing a list of protocols to be used.
- each protocol *pi* is identified by its name and a list of parameters

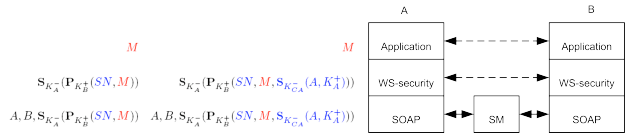
Overview of the Protocols

Go one step further and consider the various protocols needed to secure the communication.



- WS-Security (Web Service Security): a communication protocol suite providing security to Web services, while guaranteeing point-to-point (P2P) integrity and authenticity.
- SOAP (Simple Object Access Protocol): a protocol specification for exchanging structured information in computer networks.[2ex]
- TLS (Transport Layer Security): a key establishment protocol.

Description of WS-Security and SOAP Protocols



- A is sending a message M to B
- WS-Security ($P2P$ or $P2P^+$) signs the encrypted messages to ensure integrity and confidentiality
  - SN: a sequence number for correlation
  - $P2P^+$ also inserts A's certificate in the message in case it is unknown to B, to allow B to reply.
- SOAP gives a standardised format for the messages and adds the message header to allow SOAP-routing

Description of TLS Protocol

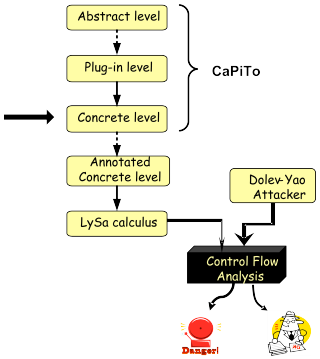
The TLS protocol is summarised by the following protocol narration taking place between the client C and a server (Bank) S holding a certificate $\mathbf{S}_{K^-_{CA}}(S, K^+_S)$ issued by a mutually trusted Certificate Authority CA:

$$
\begin{array}{llllll}
1. & C & \rightarrow & S & : & N_C \\
2. & S & \rightarrow & C & : & N_S, \mathbf{S}_{K^-_{CA}}(S, K^+_S) \\
3. & C & \rightarrow & S & : & \mathbf{P}_{K^+_S}(N) \\
4. & C & \rightarrow & S & : & \mathbf{E}_{\mathbf{H}(N_C, N_S, N)}(M)
\end{array}
$$

Protocol Stack Plug-in Level Specification

$$
\begin{array}{lll}
Client & \triangleq & !\,(\nu\ Bta)\overline{credit\_req}[TLS, Client, Bank]. \\
 & & \quad \big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\big) \\
Bank & \triangleq & !\ credit\_req[TLS, Client, Bank].(?y_{bta}). \\
 & & \quad \overline{validate}[P2P^+, Bank, Ser_E; SOAP, Bank, SM, Ser_E]. \\
 & & \quad\quad \big(\langle y_{bta}\rangle.(?y_{res}).\uparrow\langle y_{bta}, y_{res}\rangle.0\big) \\
Ser_E & \triangleq & !\ validate[P2P^+, Bank, Ser_E; SOAP, Bank, SM, Ser_E]. \\
 & & \quad \big((?z_{bta}).\langle isValid(z_{bta})\rangle.0\big) \\
SM & \triangleq & !\,(?A, SM, A, ?B, ?M).\langle SM, B, A, B, M\rangle
\end{array}
$$

Next: Introduction to Concrete Level

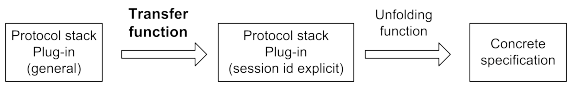


Concrete Level Syntax

- Cryptography features
  - Asymmetric encryption $\mathbf{P}_{v^+}(\vec{v})$ and decryption $\underline{\mathbf{P}}_{v^-}(\vec{v})$
  - Digital signature $\mathbf{S}_{v^-}(\vec{v})$ and verification $\underline{\mathbf{S}}_{v^+}(\vec{v})$
  - Hash function $\mathbf{H}(\vec{v})$
  - Symmetric encryption $r : e \blacktriangleright P$ with e being the symmetric key
- Processes:

$$
\begin{array}{lll}
P & ::= & \langle r, \vec{e}\rangle.P \mid (r, \vec{p}).P \mid (\nu\ n)P \mid \\
 & & !P \mid P_1 + P_2 \mid P_1 | P_2 \mid 0 \mid \\
 & & \overline{n}[\ ].P \mid n[\ ].P \mid \uparrow \langle \vec{e}\rangle.P \mid \\
 & & (\nu_{\pm}\ n)P \mid \\
 & & r : e \blacktriangleright P
\end{array}
$$

From Plug-in Level to Concrete Level (1) – A Transfer Function

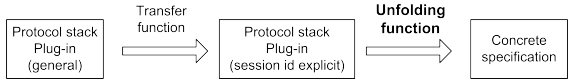


- Distinguish between different services as well as different sessions of each service by unique session identifier r .

$$
\begin{array}{lll}
\mathcal{T}(\overline{s}[pi] \cdot P, rl) & \triangleq & (\nu\ r)\langle r_{env}, s, r\rangle.\overline{r}[pi] \rhd \mathcal{T}(P, r :: rl) \\
\mathcal{T}(s[pi] \cdot P, rl) & \triangleq & (r_{env}, s, ?r).r[pi] \rhd \mathcal{T}(P, r :: rl) \\
\mathcal{T}(\langle \vec{e}\rangle.P, r :: rl) & \triangleq & \langle r, \vec{e}\rangle.\mathcal{T}(P, r :: rl) \\
\mathcal{T}((\vec{p}).P, r :: rl) & \triangleq & (r, \vec{p}).\mathcal{T}(P, r :: rl) \\
\mathcal{T}(\uparrow \langle \vec{e}\rangle.P, r_1 :: r_2 :: rl) & \triangleq & \langle r_2, \vec{e}\rangle.\mathcal{T}(P, r_1 :: r_2 :: rl)
\end{array}
$$

- Result: $s[pi] \cdot P \Rightarrow r[pi] \rhd P$

From Plug-in Level to Concrete Level (2) – An Unfolding Function



- When a protocol stack contains a sequence of protocols we first expand away the leftmost (topmost) protocol in the stack and then continue with the subsequent layers

$$
\begin{array}{lll}
\overline{r}[pi_1; \ldots; pi_k] \rhd P & \triangleq & \overline{r}[pi_k] \rhd (\ldots(\overline{r}[pi_1] \rhd P)) \\
r[pi_1; \ldots; pi_k] \rhd P & \triangleq & r[pi_k] \rhd (\ldots(r[pi_1] \rhd P))
\end{array}
$$

Narration of TLS Protocol

- Explicit about the creation of nonces and the checking of certificates
- After the completion, the symmetric key $\mathbf{H}(N_C, N_S, N)$ is computed

$$
\begin{array}{llllll}
1. & C & \rightarrow & S & : & N_C \\
2. & S & \rightarrow & C & : & N_S, \mathbf{S}_{K^-_{CA}}(S, K^+_S) \\
3. & C & \rightarrow & S & : & \mathbf{P}_{K^+_S}(N) \\
4. & C & \rightarrow & S & : & \mathbf{E}_{\mathbf{H}(N_C, N_S, N)}(M)
\end{array}
$$

Concrete Specification of TLS

$$
\begin{array}{lll}
\bar{r}[TLS, C, S, CA] \triangleright (\langle M \rangle.P) & \triangleq & (\nu\ N_C)\langle r, C, S, N_C \rangle. \\
& & \quad (r, S, C, ?n_S, \underline{\mathbf{S}}_{K^+_{CA}}(S, ?x_{ks})). \\
& & \quad\quad (\nu\ N)\langle r, C, S, \mathbf{P}_{x_{ks}}(N) \rangle. \\
& & \quad\quad\quad r : \mathbf{H}(N_C, n_S, N) \blacktriangleright (\langle M \rangle.P) \\
r[TLS, C, S, CA] \triangleright ((?m).P) & \triangleq & (r, C, S, ?n_C). \\
& & \quad (\nu\ N_S)\langle r, S, C, N_S, \mathbf{S}_{K^-_{CA}}(S, K^+_S) \rangle. \\
& & \quad\quad (r, C, S, \underline{\mathbf{P}}_{K^-_S}(?n)). \\
& & \quad\quad\quad r : \mathbf{H}(n_C, N_S, n) \blacktriangleright ((?m).P)
\end{array}
$$

parameterised on the session identifier r

Concrete Specification of TLS

$$
\begin{array}{lll}
\bar{r}[TLS, C, S, CA] \triangleright (\langle M \rangle.P) & \triangleq & (\nu\ N_C)\langle r, C, S, N_C \rangle. \\
& & \quad (r, S, C, ?n_S, \underline{\mathbf{S}}_{K^+_{CA}}(S, ?x_{ks})). \\
& & \quad\quad (\nu\ N)\langle r, C, S, \mathbf{P}_{x_{ks}}(N) \rangle. \\
& & \quad\quad\quad r : \mathbf{H}(N_C, n_S, N) \blacktriangleright (\langle M \rangle.P) \\
r[TLS, C, S, CA] \triangleright ((?m).P) & \triangleq & (r, C, S, ?n_C). \\
& & \quad (\nu\ N_S)\langle r, S, C, N_S, \mathbf{S}_{K^-_{CA}}(S, K^+_S) \rangle. \\
& & \quad\quad (r, C, S, \underline{\mathbf{P}}_{K^-_S}(?n)). \\
& & \quad\quad\quad r : \mathbf{H}(n_C, N_S, n) \blacktriangleright ((?m).P)
\end{array}
$$

parameterised on the session identifier r

Concrete Specification of TLS

$$
\begin{array}{lll}
\bar{r}[TLS, C, S, CA] \triangleright (\langle M \rangle.P) & \triangleq & (\nu\ N_C)\langle r, C, S, N_C \rangle. \\
& & \quad (r, S, C, ?n_S, \underline{\mathbf{S}}_{K^+_{CA}}(S, ?x_{ks})). \\
& & \quad\quad (\nu\ N)\langle r, C, S, \mathbf{P}_{x_{ks}}(N) \rangle. \\
& & \quad\quad\quad r : \mathbf{H}(N_C, n_S, N) \blacktriangleright (\langle M \rangle.P) \\
r[TLS, C, S, CA] \triangleright ((?m).P) & \triangleq & (r, C, S, ?n_C). \\
& & \quad (\nu\ N_S)\langle r, S, C, N_S, \mathbf{S}_{K^-_{CA}}(S, K^+_S) \rangle. \\
& & \quad\quad (r, C, S, \underline{\mathbf{P}}_{K^-_S}(?n)). \\
& & \quad\quad\quad r : \mathbf{H}(n_C, N_S, n) \blacktriangleright ((?m).P)
\end{array}
$$

parameterised on the session identifier r

Concrete Specification of TLS

$$\bar{r}[TLS, C, S, CA] \triangleright (\langle M \rangle.P) \triangleq (\nu\ N_C)\langle r, C, S, N_C \rangle.(r, S, C, ?n_S, \underline{\mathbf{S}}_{K^+_{CA}}(S, ?x_{ks})).(\nu\ N)\langle r, C, S, \mathbf{P}_{x_{ks}}(N)\rangle.\ r : \mathbf{H}(N_C, n_S, N) \blacktriangleright (\langle M \rangle.P)$$

$$r[TLS, C, S, CA] \triangleright ((?m).P) \triangleq (r, C, S, ?n_C).(\nu\ N_S)\langle r, S, C, N_S, \mathbf{S}_{K^-_{CA}}(S, K^+_S)\rangle.(r, C, S, \underline{\mathbf{P}}_{K^-_S}(?n)).\ r : \mathbf{H}(n_C, N_S, n) \blacktriangleright ((?m).P)$$

parameterised on the session identifier r

Concrete Specification of WS-Security

Sender sends out message M: $A \rightarrow B : M$.

- $P2P^+$: encrypts and signs the message together with a sequence number SN and sender's certificate $\mathbf{S}_{K^-_{CA}}(A, K^+_A)$ to allow receiver to reply.

$$\bar{r}[P2P^+, S, R, CA] \triangleright (\langle r, \vec{e} \rangle.P) \triangleq (\nu\ SN)\langle r, \mathbf{S}_{K^-_A}(\mathbf{P}_{K^+_B}(SN, \vec{e}, \mathbf{S}_{K^-_{CA}}(A, K^+_A)))\rangle.\bar{r}[P2P, S, R, SN] \triangleright P$$

- $P2P$: encrypts and signs the message together with a sequence number SN.

$$\bar{r}[P2P, S, R, SN] \triangleright (\langle r, \vec{e} \rangle.P) \triangleq \langle r, \mathbf{S}_{K^-_A}(\mathbf{P}_{K^+_B}(SN, \vec{e}))\rangle.\bar{r}[P2P, S, R, SN] \triangleright P$$

Concrete Specification of *SOAP*

- SOAP: appends additional fields (headers) to messages.

$$\bar{r}[SOAP, S, SM, R] \triangleright (\langle r, \vec{e} \rangle.P) \triangleq \langle r, S, SM, S, R, \vec{e} \rangle.\bar{r}[SOAP, S, SM, R] \triangleright P$$

$$\bar{r}[SOAP, S, SM, R] \triangleright ((r, \vec{p}).P) \triangleq (r, SM, S, R, S, \vec{p}).\bar{r}[SOAP, S, SM, R] \triangleright P$$

Example: Transformations of Bank

Protocol stack plug-in specification of the Bank:

$$Bank \triangleq\ !\ credit\_req[TLS, Client, Bank].(?y_{bta}).\overline{validate}[WS, Bank, Ser_E; SOAP, Bank, SM, Ser_E].\left(\langle y_{bta} \rangle.(?y_{res}).\uparrow \langle y_{bta}, y_{res} \rangle.0\right)$$

We shall show how it is transformed into concrete level specification.

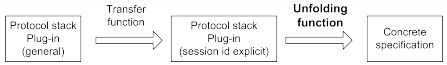
Example: Applying the Transfer Function

$$Bank \triangleq\ !\ \overline{credit\_req}[TLS, Client, Bank].(?y_{bta}).\ \overline{validate}[WS, Bank, Ser_E; SOAP, Bank, SM, Ser_E].\ \big(\langle y_{bta}\rangle.(?y_{res}).\uparrow\langle y_{bta}, y_{res}\rangle.0\big)$$

$$\Downarrow$$

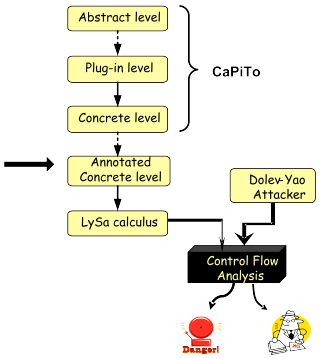
$$\mathcal{T}(Bank, [r_{env}]) \triangleq (r_{env}, req, ?r_1).r_1[TLS, Client, Bank] \triangleright \big((r_1, ?y_{bta}).\ (\nu\ r_2)\langle r_{env}, val, r_2\rangle.\ \overline{r_2}[P2P^+, Bank, Ser_E; SOAP, Bank, SM, Ser_E] \triangleright \big(\langle r_2, y_{bta}\rangle.(r_2, ?y_r).\langle r_1, y_{bta}, y_r\rangle.0\big)\big)$$

Example: Applying the Unfolding Function



$$(r_{env}, req, ?r_1).(r_1, Client, Bank, ?n_C).\ (\nu\ N_S)\langle r_1, Bank, Client, N_S, \mathbf{S}_{K^-_{CA}}(Bank, K^+_{Bank})\rangle\ (r_1, Client, Bank, \underline{\mathbf{P}}_{K^-_{Bank}}(?n)).\ r_1 : \mathbf{H}(n_C, N_S, n) \blacktriangleright \big((r_1, ?y_{bta}).\ (\nu\ r_2)\langle r_{env}, val, r_2\rangle.(\nu\ SN)\ \langle r_2, Bank, SM, Bank, Ser_E, \mathbf{S}_{K^-_{Bank}}(\mathbf{P}_{K^+_{Ser_E}}(SN, y_{bta}, \mathbf{S}_{K^-_{CA}}(Bank, K^+_{Bank})))\rangle.\ (r_2, SM, Bank, Ser_E, Bank, \underline{\mathbf{S}}_{K^+_{Ser_E}}(\underline{\mathbf{P}}_{K^-_{Bank}}(SN, ?y_r))).\ \langle r_1, y_{bta}, y_r\rangle.0\big)$$

Next: Introduction of Annotated Concrete Level



Authentication Annotations

Focus on encryptions (rather than communication)

- when they are created, specify where they are intended to be decrypted

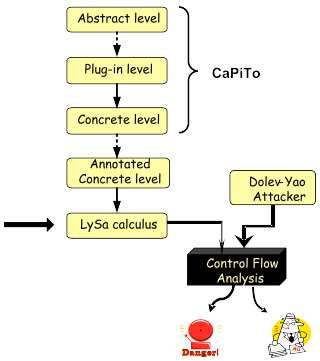
$$r : e \blacktriangleright \langle V_1, V_2\rangle^{l_1}[dest\ \mathcal{L}_2].P$$

- when they are decrypted, specify where they are expected to have been encrypted

$$r : e \blacktriangleright (V_1, x)^{l_2}[orig\ \mathcal{L}_1].P$$

l_1 and l_2 are crypto-points. $\mathcal{L}_1$ and $\mathcal{L}_2$ are sets of crypto-points.
Ideally, $l_1 \in \mathcal{L}_2$ and $l_2 \in \mathcal{L}_1$
Similar annotations are made to asymmetric encryption, signature, decryption and signature validation.

Next: Introduction to LySa Calculus



The LySa Calculus

- Expressions

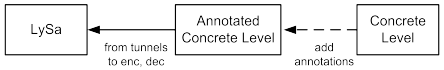
$$\begin{array}{rcl} E & ::= & n \\ & | & x \\ & | & \{E_1, \ldots, E_k\}^{l}_{E_0}[dest\ \mathcal{L}] \end{array}$$

- Processes

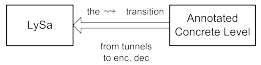
$$\begin{array}{rcl} P & ::= & \langle E_1, \ldots, E_k \rangle.P \\ & | & (E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P \\ & | & \text{decrypt } E \text{ as } \{E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\}^{l}_{E_0}[orig\ \mathcal{L}] \text{ in } P \\ & | & (\nu\ n)P \\ & | & P_1 | P_2 \\ & | & !P \\ & | & 0 \end{array}$$

From CaPiTo to LySa

- Tool: LySa-Tool is an analysis tool to analyse protocols (applications) specified in process algebra LySa.
- Goal: use LySa tool to analyse concrete specifications of Service-Oriented application.
- Mean: translate CaPiTo into LySa.
  - Add annotations
  - Transform tunnels into encryptions and decryptions



From Annotated Concrete Level to LySa – The $\rightsquigarrow$ Transition



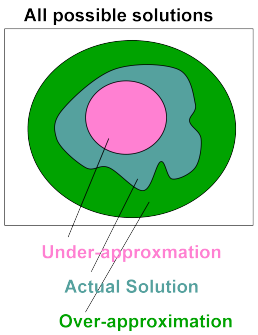
- LySa: encryptions ($\{M\}_K$) and decryptions (decrypt V as $\{; m\}_K$)
- CaPiTo: tunnels ($r : e \blacktriangleright P$)

$$r : e_0 \blacktriangleright \langle r', \vec{e} \rangle^{l_0}[dest\ \mathcal{L}].P \rightsquigarrow \langle r', \{\vec{e}\}^{l_0}_{e_0}[dest\ \mathcal{L}] \rangle.r : e_0 \blacktriangleright P \quad \text{if } r = r'$$
$$r : e_0 \blacktriangleright \langle r', \vec{e} \rangle^{l_0}[dest\ \mathcal{L}].P \rightsquigarrow \langle r', \vec{e} \rangle^{l_0}[dest\ \mathcal{L}].r : e_0 \blacktriangleright P \quad \text{if } r \neq r'$$

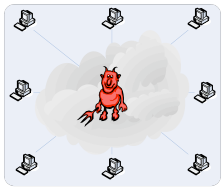
$$r : e_0 \blacktriangleright (r', \vec{p})^{l_0}[orig\ \mathcal{L}].P \rightsquigarrow (r', \{\vec{p}\}^{l_0}_{e_0}[orig\ \mathcal{L}]).r : e_0 \blacktriangleright P$$
$$\text{if } r = r' \text{ and } \mathrm{fn}(e_0) \cap \mathrm{fn}(\vec{p}) = \emptyset$$
$$r : e_0 \blacktriangleright (r', \vec{p})^{l_0}[orig\ \mathcal{L}].P \rightsquigarrow (r', \vec{p})^{l_0}[orig\ \mathcal{L}].r : e_0 \blacktriangleright P \quad \text{if } r \neq r'$$

Static Analysis

- Properties of Static Program Analysis
  - Semantically correct
    - Over-Approximation (e.g. this work)
    - Under-Approximation (e.g. AVISPA)
- Efficient algorithms



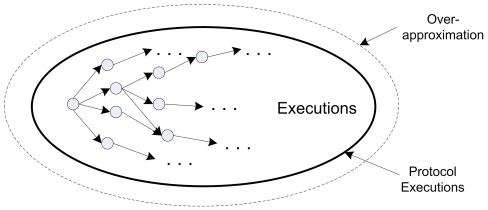
The Dolev-Yao Attacker



- Challenge: the existence of the attacker
- Capabilities
  - receive and send messages on the network
  - encrypt and decrypt messages using known keys
  - create new keys, nonces, messages, etc
- Assumption: Prefect cryptography

The Analysis of LySa

The analysis collects (over-approximates) information of LySa process executions.



The Analysis Specifications

The analysis components contain the information of executions.

- notations: *Val*: values (ground terms), *Var*: variables, and *Lab*: labels (crypto-points)
- $\rho : \mathcal{P}(Var \times Val)$ records for each variable the set of names it may be bound to.
- $\kappa : \mathcal{P}(Val^*)$ records all the tuples that has been sent to a server.
- $\psi : \mathcal{P}(Lab \times Lab)$ contains an over-approximation of the potential origin/destination violations. If $(l, l') \in \psi$ then something encrypted at crypto-point l may unexpectedly be decrypted at crypto-point l', or something decrypted at l' was expected to be encrypted at another place than l

The analysis checks at each decryption whether any violation may happen and collect it in the error component.

The Judgements

- The judgement for analysing expressions takes the form

$$\rho \models e : \vartheta$$

  - demanding that ϑ contains all the values associated with the components of an expression.
- The judgement for analysing processes takes the form

$$\rho, \kappa \models P : \psi$$

  - for each process P, collects information into ρ, κ and annotation violations ψ.

Analyse Expressions

Analysis Rules of Expressions

(name) $\rho \models n : \vartheta$ iff $n \in \vartheta$

(variable) $\rho \models x : \vartheta$ iff $\rho(x) \subseteq \vartheta$

(encry) $\rho \models \mathbf{P}_{v_0^+}(v_1, \ldots, v_k)^l[dest\ \mathcal{L}] : \vartheta$

$$\begin{array}{l} \text{iff } \wedge_{i=0}^{k} \rho \models v_i : \vartheta_i \wedge \\ \quad \forall w_0, w_1, \ldots, w_k : \wedge_{i=0}^{k} w_i \in \vartheta_i \Rightarrow \\ \quad\quad \mathbf{P}_{w_0^+}(w_1, \ldots, w_k)^l[dest\ \mathcal{L}] \in \vartheta \end{array}$$

Analyse Processes (1)

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

$$\frac{\rho, \kappa \models P_1 : \psi \wedge \rho, \kappa \models P_2 : \psi}{\rho, \kappa \models P_1 | P_2 : \psi}$$

$$\frac{\rho, \kappa \models P : \psi}{\rho, \kappa \models !P : \psi}$$

Analyse Processes (2)

Analysis Rules of Processes

(output) $\rho, \kappa \models \langle e_1, \ldots, e_k \rangle.P : \psi$

$$\begin{array}{l} \text{iff } \wedge_{i=1}^{k} \rho \models e_i : \vartheta_i \wedge \\ \quad \forall w_1, \ldots, w_k : \wedge_{i=1}^{k} w_i \in \vartheta_i \Rightarrow \langle w_1, \ldots, w_k \rangle \in \kappa \wedge \\ \quad\quad \rho, \kappa \models P : \psi \end{array}$$

(input) $\rho, \kappa \models (p_1, \ldots, p_k).P : \psi$

$$\begin{array}{l} \text{iff } \rho \models_1 p_1, \ldots, p_k : \kappa \triangleright \hat{W} : \psi \wedge \\ \quad \hat{W} \neq \emptyset \Rightarrow \rho, \kappa \models P : \psi \end{array}$$

$\rho \models_1 p_1, \ldots, p_k : \kappa \triangleright \hat{W} : \psi$ is a judgement for pattern matching, which selects tuples from κ that match $p_1, \ldots, p_k$ into $\hat{W}$.

Pattern Matching

The general judgement for pattern matching

$$\rho \models_i \vec{p} : \hat{V} \triangleright \hat{W} : \psi$$

Tuples in $\hat{V}$ that satisfy the requirements are collected into $\hat{W}$. The judgement also collects *orig* and *dest* annotation violations into ψ. For example:

$$\rho \models_1 (r, A, ?m) : \{(r, A, M), (r, B, M)\} \triangleright \{(r, A, M)\} : \emptyset$$

holds with variable m becoming bound to value M on the fly, and

$$\rho \models_1 \{A, ?m\}^{l_1}_K[orig\ \{l_2\}] : \{\{A, M\}^{l_3}_K[dest\ \{l_4\}], (B, M)\} \triangleright \{\{A, M\}_K\} : \{(l_1, l_3)\}$$

holds with the annotation violation (l_1, l_3) being recorded in ψ (because $l_1 \notin \{l_4\}$ and $l_3 \notin \{l_2\}$).

Construct Constraints

The analysis gives rise to a set of constraints, in the form of logical formulae, that are to be solved.

$$\begin{array}{ll} (\nu\ Key) & (\\ & (\nu\ Res)\langle A, B, \{Res\}^{l_1}_{Key}[dest\ \{l_2\}]\rangle.0 \\ & \mid\ (A, B; x).\ \text{decrypt}\ x\ \text{as}\ \{; x_{res}\}^{l_2}_{Key}[orig\ \{l_1\}]\ \text{in}\ 0) \end{array}$$

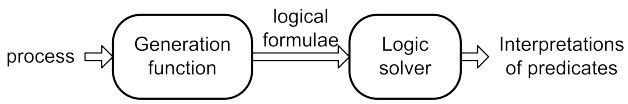
The constraints generated include:

$$\langle A, B, \{Res\}^{l_1}_{Key}[dest\ \{l_2\}]\rangle \in \kappa$$

$$\forall v : \langle A, B, v\rangle \in \kappa \Rightarrow v \in \rho(x)$$

$$\forall v, l, \mathcal{L} : \{v\}^{l}_{Key}[dest\ \mathcal{L}] \in \rho(x) \Rightarrow v \in \rho(x_{res}) \wedge (\neg(l \in \{l_1\}) \vee \neg(l_2 \in \mathcal{L}) \Rightarrow (l, l_2) \in \psi)$$

Implementation Outline



- We adopt Succinct Solver as the Logic Solver.
- The results are computed in low polynomial time in the size of the system. In practice, less than two seconds.
- Alternatively, there are other candidates for the position of Logic Solver.
  - Datalog
  - H1, H3, ...

Implementation Detail – From Infinite to Finite

The analysis is specified over the *infinite* set of value, e.g. arbitrarily deeply nested encryptions. For example,

$$\langle n\rangle.0 \mid !(; x).\langle\{x\}_k\rangle.0$$

This problem is addressed by two steps,

- Adding labels to expressions, e.g.

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

- Introducing *tree grammar* γ to present sets of values that a labelled expression may be evaluated to.

$$\rho(x) = \{l_1, l_4\} \qquad \gamma : \begin{array}{l} l_1 \to n \\ l_2 \to n \qquad l_2 \to \{l_2\}_{l_3} \\ l_3 \to k \\ l_4 \to \{l_2\}_{l_3} \end{array}$$

$$\kappa = \{l_1, l_4\}$$

Analyse the Case Study

The overall scenario of the case study takes the form

$$((\nu_{\pm}\ K_{VS})(\nu_{\pm}\ K_{Ser_E})(\nu_{\pm}\ K_{Ser_C}) \\ |_{i=1}^{n}\ Client_i \mid VS \mid SM \mid Ser_E \mid Ser_C) \\ \mid\ Attacker$$

The Scenario:

- i Clients may simultaneously request services from VS and Ser_E (or Ser_C). The index i is added to all variables, crypto-points and constants
- An active Dolev-Yao attacker. The attacker has no knowledge of the principals' asymmetric keys.

In our experiment, the analysis is carried out for $n = 2$, thus allows the analysis to see if the two instance of communications can interfere with each other.

Analysis Results

$$\begin{aligned} Client &\triangleq\ !\ (\nu\ Bta)\overline{credit\_req}[\,]\cdot\Big(\langle Bta\rangle.(Bta, ?x_{res}).\uparrow\langle x_{res}\rangle.0\Big) \\ Bank &\triangleq\ !\ credit\_req[\,]\cdot\Big((?y_{bta}).\overline{validate}[\,]\cdot\langle y_{bta}\rangle.(?y_{res}).\uparrow\langle y_{bta}.y_{res}\rangle.0\Big) \\ Ser_E &\triangleq\ !\ validate[\,]\cdot\Big((?z_{bta}).\langle isValid(z_{bta})\rangle.0\Big) \end{aligned}$$

Analysing the case study gives an empty ψ component

- no authentication annotations are violated

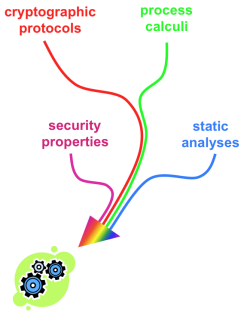
Some entries may be of interest:

$$\boxed{\begin{aligned}\rho(x_{res1}) &= \{isValid(Bta_1)\} \\ \rho(x_{res2}) &= \{isValid(Bta_2)\}\end{aligned}}$$

- Confirms that the evaluation results $isValid(Bta_i)$ are correctly returned back from Ser_E to $Client$, via VS and SM

LySa Status[1] – Applications

- LySa tool has been applied to verify a number of security protocols
  - Classical protocol, e.g. Needham-Schroeder, Otway-Rees
  - Modern protocols, e.g. Electronic voting protocols, SAML Single Sign-On protocol



[1] LySa tool: http://www2.imm.dtu.dk/cs_LySa/lysatool/

Analyse Classical Protocols

protocol $1 \le i,j \le n,\ i \ne j$	$A \ne B$ $\wedge_{i=0}^{n} K_i^A \ne K_i^B$	$A = B$ $\wedge_{i=0}^{n} K_i^A \ne K_i^B$	$A \ne B$ $\wedge_{i=0}^{n} K_i^A = K_i^B$	$A = B$ $\wedge_{i=0}^{n} K_i^A = K_i^B$
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 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 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 flaw corrected	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

- $A \ne B$: distinguish between roles
- $K_A \ne K_B$: different master keys for the two roles

Electronic Voting Protocols

- Convenient and inexpensive
- Several cryptographic approaches
- Introduces new ways to disrupt or falsify votings
- Must upheld the security properties of the classical paper voting
- Need for provably correct systems

Security Properties

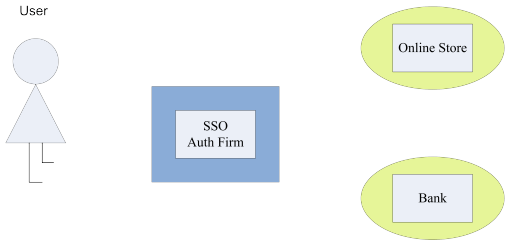
- Verifiability
  - Voters can verify that their votes have been counted
- Accuracy
  - No votes can be altered
  - Validated votes count in the final tally
  - Invalid votes cannot be counted in the final tally
- Democracy
  - Only eligible voters can vote
  - Eligible voters can only vote once
- Fairness
  - No early results from the voting can be obtained

Analyse Electronic Voting Protocols

	FOO92	FOO92 Amended	Sensus	Sensus Amended	E-Vox
Verifiability	No	Yes	Yes	Yes	Yes
Accuracy	No	Yes	No	Yes	Yes
Democracy	Yes	Yes	Yes	Yes	Yes
Fairness	Yes	Yes	Yes	Yes	No

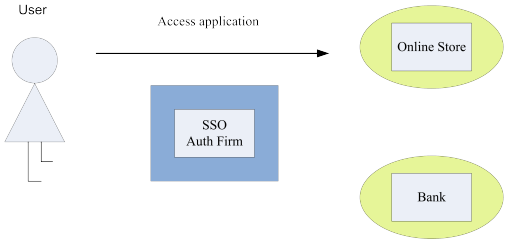
SAML Single Sign-On Protocol

User may access to multiple applications with login once



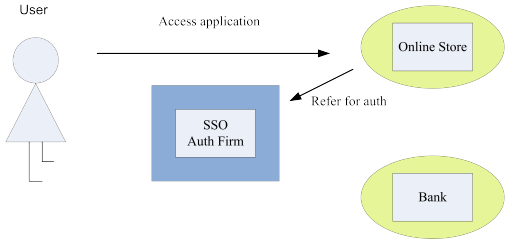
SAML Single Sign-On Protocol

User may access to multiple applications with login once



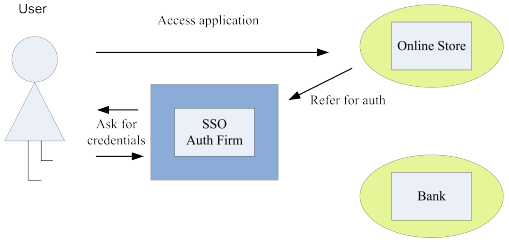
SAML Single Sign-On Protocol

User may access to multiple applications with login once



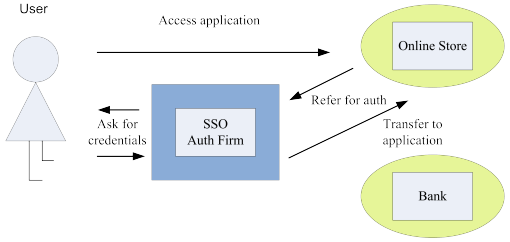
SAML Single Sign-On Protocol

User may access to multiple applications with login once



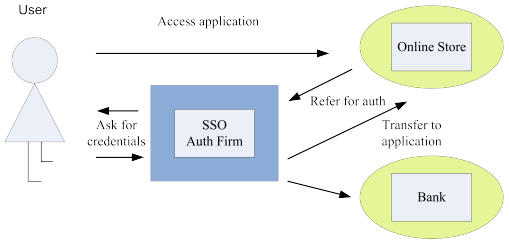
SAML Single Sign-On Protocol

User may access to multiple applications with login once



SAML Single Sign-On Protocol

User may access to multiple applications with login once



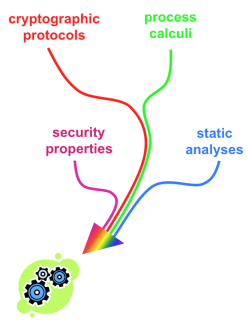
Analyse the SAML Single Sign-On Protocol

Ex #	Scenario	Description	$\rho(\mathbf{z}_\bullet)$	ψ
1		Multiple Errors	$Mess \in \rho(z_\bullet)$	$\psi \neq \emptyset$
2		User gets arbitrary result	$Mess \notin \rho(z_\bullet)$	$\psi \neq \emptyset$
3		No Errors	$Mess \notin \rho(z_\bullet)$	$\psi = \emptyset$
4		Compromised $Mess$	$Mess \in \rho(z_\bullet)$	$\psi \neq \emptyset$
5		No Errors	$Mess \notin \rho(z_\bullet)$	$\psi = \emptyset$
6		No Errors	$Mess \notin \rho(z_\bullet)$	$\psi = \emptyset$
7		User gets arbitrary result	$Mess \notin \rho(z_\bullet)$	$\psi \neq \emptyset$
8		No Errors	$Mess \notin \rho(z_\bullet)$	$\psi = \emptyset$

- U: user, D: application, S: authentication firm
- $\dashrightarrow$: messages are sent without protected by TLS, $\longrightarrow$: messages are protected by TLS
- principals are authenticated when pointed by arrows

LySa Status[2] – Additional Features

- LySa tool has been applied to verify a number of security protocols
  - Classical protocol, e.g. Needham-Schroeder, Otway-Rees
  - Modern protocols, e.g. Electronic voting protocols, SAML Single Sign-On protocol
- Extensions of LySa have been developed and are still being developed
  - New security properties
  - New primitives in LySa
  - New type of protocols
  - More powerful analysis



[2] LySa tool: http://www2.imm.dtu.dk/cs_LySa/lysatool/

Additional Security Properties

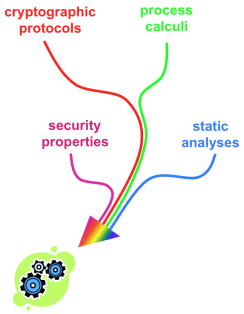
LySa tool has been extended to deal with more security properties

- Confidentiality: messages are not leaked to a party (or the attacker) that is not suppose to know
- Freshness: messages are fresh to the current session
- Absence of simple type flaws: a field of one type is not confused with a field of another type
- Absence of complex type flaws: a concatenation of fields are not confused with a single field

The extensions are based on the same technique, namely, expressing the intension of protocols by means of annotations and applying the analysis to capture any possible violations.

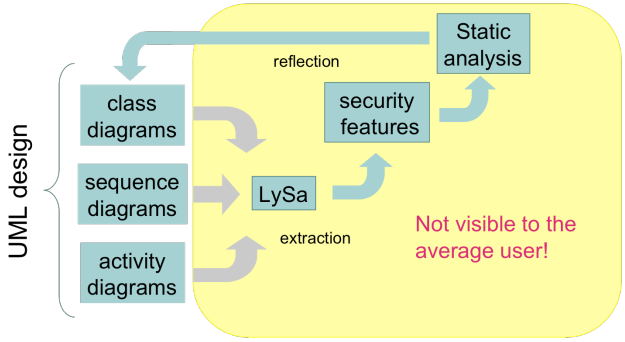
LySa Status[3] – Tool Sets

- LySa tool has been applied to verify a number of security protocols
  - Classical protocol, e.g. Needham-Schroeder, Otway-Rees
  - Modern protocols, e.g. Electronic voting protocols, SAML Single Sign-On protocol
- Extensions of LySa have been developed and are still being developed
  - New security properties
  - New primitives in LySa
  - New type of protocols
  - More powerful analysis
- Tool sets to LySa



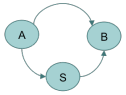
[3] LySa tool: http://www2.imm.dtu.dk/cs_LySa/lysatool/

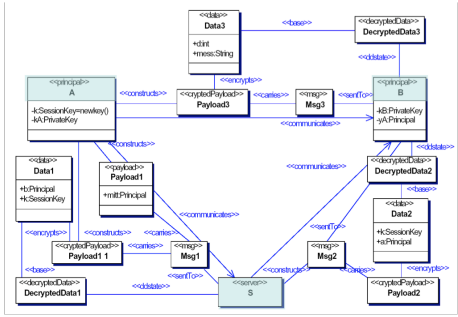
Integration in Software Engineering Tools



UML Replacing LySa (1)

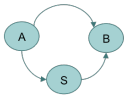
Class diagram for WMF



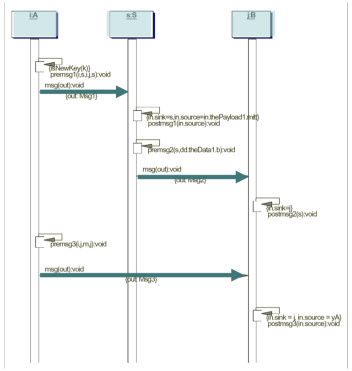


UML Replacing LySa (2)

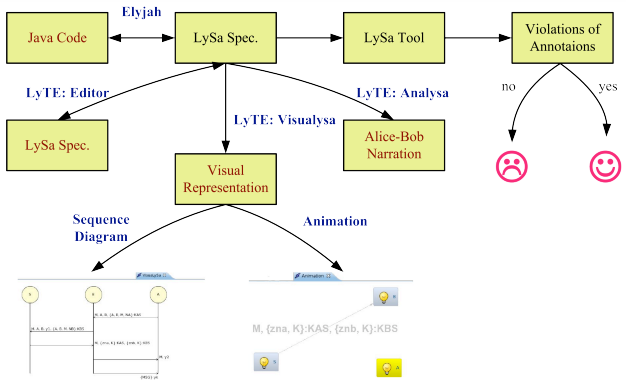
Sequence diagram for WMF



Each diagram only explains part of what happens — additional diagrams enhance our understanding

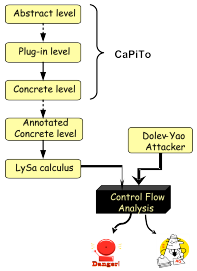


Elyjah and LyTE Tools[4]



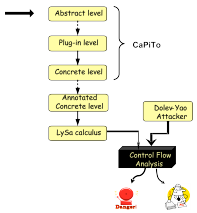
[4] Elyjah & LyTE: http://homepages.inf.ed.ac.uk/s0237477/

Summary



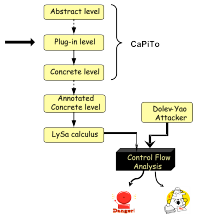
- The CaPiTo approach to model service oriented systems at different levels of abstraction
  - The abstract level allows us to concentrate on the application itself
  - The protocol stack plug-in level is precise about protocols and connects the abstract and concrete level
  - The concrete level allows us to take the underlying protocols into consideration
- The annotated concrete level formalises the intentions of protocols
- The transformation from annotated concrete level to LySa allows us to reuse the static analyser – LySa tool, which tracks the run-time behaviour and checks the authentication properties of systems

Summary



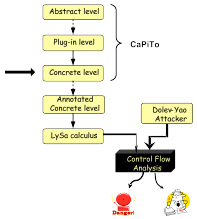
- The CaPiTo approach to model service oriented systems at different levels of abstraction
  - The abstract level allows us to concentrate on the application itself
  - The protocol stack plug-in level is precise about protocols and connects the abstract and concrete level
  - The concrete level allows us to take the underlying protocols into consideration
- The annotated concrete level formalises the intentions of protocols
- The transformation from annotated concrete level to LySa allows us to reuse the static analyser – LySa tool, which tracks the run-time behaviour and checks the authentication properties of systems

Summary



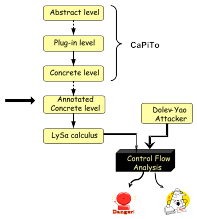
- The CaPiTo approach to model service oriented systems at different levels of abstraction
  - The abstract level allows us to concentrate on the application itself
  - The protocol stack plug-in level is precise about protocols and connects the abstract and concrete level
  - The concrete level allows us to take the underlying protocols into consideration
- The annotated concrete level formalises the intentions of protocols
- The transformation from annotated concrete level to LySa allows us to reuse the static analyser – LySa tool, which tracks the run-time behaviour and checks the authentication properties of systems

Summary



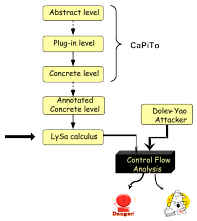
- The CaPiTo approach to model service oriented systems at different levels of abstraction
  - The abstract level allows us to concentrate on the application itself
  - The protocol stack plug-in level is precise about protocols and connects the abstract and concrete level
  - The concrete level allows us to take the underlying protocols into consideration
- The annotated concrete level formalises the intentions of protocols
- The transformation from annotated concrete level to LySa allows us to reuse the static analyser – LySa tool, which tracks the run-time behaviour and checks the authentication properties of systems

Summary



- The CaPiTo approach to model service oriented systems at different levels of abstraction
  - The abstract level allows us to concentrate on the application itself
  - The protocol stack plug-in level is precise about protocols and connects the abstract and concrete level
  - The concrete level allows us to take the underlying protocols into consideration
- The annotated concrete level formalises the intentions of protocols
- The transformation from annotated concrete level to LySa allows us to reuse the static analyser – LySa tool, which tracks the run-time behaviour and checks the authentication properties of systems

Summary



- The CaPiTo approach to model service oriented systems at different levels of abstraction
  - The abstract level allows us to concentrate on the application itself
  - The protocol stack plug-in level is precise about protocols and connects the abstract and concrete level
  - The concrete level allows us to take the underlying protocols into consideration
- The annotated concrete level formalises the intentions of protocols
- The transformation from annotated concrete level to LySa allows us to reuse the static analyser – LySa tool, which tracks the run-time behaviour and checks the authentication properties of systems

Contribution

- allows to perform an abstract modelling of service-oriented applications
- facilitates dealing with existing industrial protocols
- bridges the gap between the academic parters and industrial partners