Protocol Stacks for Services

Han Gao    Flemming Nielson    Hanne Riis Nielson

Technical University of Denmark

Workshop on Formal Methods for Security
Padova, March 13th, 2009
Service Oriented Architecture and Protocols

- Definition: A service-oriented architecture can be defined as a group of services, which communicate with each other. The process of communication involves either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

- A service may often engage in multiple concurrent conversations to a number of services.

- Conversations may be protected by the use of different cryptographic protocols.
  - or a number of protocols running on top of each other.

- OSI model
The OSI Model

The Open Systems Interconnection Reference Model is an abstract description for layered communications and computer network protocol design. It divides network architecture into seven layers.

Different people may have different focuses.
Our Approach

We model Service-Oriented systems at different levels

- Abstract level (as often found in academia)
- Concrete level (as often found in industry)

We then show how to bridge the cap between the two levels
RoadMap

- The CaPiTo approach
- Credit Request Case Study
- Static Analysis of CaPiTo
- Conclusion
Abstract Specifications

At the abstract level, we abstract away from both cryptography and industrial communication protocols, and this allows us to concentrate on the interaction of the services themselves.

Syntax

\[
\begin{align*}
\text{value} & : \quad v, w \quad ::= \quad n \mid f(\vec{v}) \\
\text{expr} & : \quad e \quad ::= \quad x \mid n \mid f(\vec{e}) \\
\text{pattern} & : \quad p \quad ::= \quad ?x \mid x \mid n \\
\text{Process} & : \quad P \quad ::= \quad \langle \vec{e} \rangle.P \mid (\vec{p}).P \\
& \quad \mid (\nu \ n)P \mid P_1P_2 \mid !P \mid P_1 + P_2 \mid 0 \\
& \quad \mid \overline{n}[\ ]P \mid n[\ ]P \mid \uparrow\langle \vec{e} \rangle.P
\end{align*}
\]

Distinguish between defining occurrences and applied occurrences of variables.
A Running Example

A client asks his or her bank for the account balance

\[
\begin{align*}
\text{Client} &\rightarrow \text{Bank} : \ TS \\
\text{Bank} &\rightarrow \text{Client} : \ TS, Bal
\end{align*}
\]

In the service-oriented setup of CaPiTo, the bank will provide a service denoted \( \text{req}[\ ] \) (for request) and it is invoked by the construct \( \overline{\text{req}}[\ ] \). The client and the bank can now be specified as follows:

\[
\begin{align*}
\text{Client} \triangleq (\nu \ TS) \overline{\text{req}}[\ ]. \langle \text{Client}, \text{Bank}, TS \rangle. \\
&\quad (\text{Bank}, \text{Client}, TS, ?bal).0
\end{align*}
\]

\[
\begin{align*}
\text{Bank} \triangleq \text{req}[\ ]. (\text{Client}, \text{Bank}, ?ts). \\
&\quad (\nu \ Bal)\langle \text{Bank}, \text{Client}, ts, Bal \rangle.0
\end{align*}
\]

The overall system is the parallel composition of the above processes as given by

\[
\text{System} \triangleq \text{Client} \mid \text{Bank}
\]
Protocol Stack Plug-Ins

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

\[
\begin{align*}
\nu, \omega & ::= n \mid f(\vec{v}) \\
\varepsilon & ::= x \mid n \mid f(\vec{e}) \\
p & ::= ?x \mid x \mid n \\
\mathcal{P} & ::= \langle \varepsilon \rangle \mathcal{P} \mid (\bar{p}).\mathcal{P} \\
& \quad \mid (\nu \ n)\mathcal{P} \mid \mathcal{P}_1|\mathcal{P}_2 \mid !\mathcal{P} \mid \mathcal{P}_1 + \mathcal{P}_2 \mid 0 \\
& \quad \mid \mathcal{P}[\text{ps}] \mathcal{P} \mid n[\text{ps}] \mathcal{P} \mid \uparrow \langle \overline{\varepsilon} \rangle \mathcal{P} \\
\text{ps} & ::= \text{pi} \mid \text{pi}; \text{ps} \\
\text{pi} & ::= \text{name}, \text{param}_1, \ldots, \text{param}_k
\end{align*}
\]

- \text{ps} is a protocol stack containing a list of protocols to be used.
- each protocol \text{pi} is identified by its name and a list of parameters
Running Example Revisited

Assume TLS is used to protect the communications in the Client-Bank example.

\[
\begin{align*}
\text{Client} & \rightarrow \text{Bank} : \text{TS} \\
\text{Bank} & \rightarrow \text{Client} : \text{TS, Bal}
\end{align*}
\]

The unilateral TLS protocol is summarised by the following protocol narration taking place between a client C and a server S holding a certificate \( S_{K_{CA}}^{-}(S, K_S^+) \) issued by a mutually trusted Certificate Authority CA:

1. \( C \rightarrow S : N_C \)
2. \( S \rightarrow C : N_S, S_{K_{CA}}^{-}(S, K_S^+) \)
3. \( C \rightarrow S : P_{K_S^+}(N) \)
4. \( C \rightarrow S : E_H(N_C, N_S, N)(M) \)
The overall system will be specified as follows

\[ \text{System} \triangleq (\nu_{\pm} K_{CA})(\nu_{\pm} K_{Bank}) \ (\text{Client} \mid \text{Bank}) \]

where the previous definitions of \text{Client} and \text{Bank} are modified by “plugging in” the parameters to the service invocations and responses as shown below:

\[ \text{Client} \triangleq (\nu \ TS) \ \overline{\text{req}[\text{TLS}, \text{Client}, \text{Bank}]} \cdot \langle \text{Client}, \text{Bank}, TS \rangle \cdot (\nu \ Bal) \langle \text{Bank}, \text{Client}, ts, Bal \rangle.0 \]

\[ \text{Bank} \triangleq \text{req}[\text{TLS}, \text{Client}, \text{Bank}]. \ (\text{Client}, \text{Bank}, ?ts). \ (\nu \ Bal) \langle \text{Bank}, \text{Client}, ts, Bal \rangle.0 \]
**Concrete Specifications**

- Cryptography features
  - Asymmetric encryption $P_{v^+}(\vec{v})$ and decryption $P_{v^-}(\vec{v})$
  - Digital signature $S_{v^-}(\vec{v})$ and verification $S_{v^+}(\vec{v})$
  - Hash function $H(\vec{v})$

- Processes:

$$P ::= \langle \vec{e} \rangle .P \mid (\vec{p}) .P \mid (\nu \ n)P \mid !P \mid P_1 + P_2 \mid P_1 | P_2 \mid 0 \mid \bar{n}[ ] .P \mid n[ ] .P \mid \uparrow \langle \vec{e} \rangle .P \mid (\nu \pm n)P \mid e \triangleright P$$
Concrete Specification of TLS

- parameterised on the name $s$ of the service to be protected by the protocol stack
- explicit about the creation of nonces and the checking of certificates
- after the completion, the symmetric key $H(N_C, N_S, N)$ is computed

\[
\bar{s}[TLS, C, S].(\langle M \rangle.P) \triangleq (\nu N_C)\langle C, S, N_C \rangle.
\]
\[
\quad (S, C, ?n_S, S_{K_{CA}^+}^+(S, K_S^+))
\]
\[
\quad (\nu N)\langle C, S, P_{K_{CA}^+}^+(N) \rangle.
\]
\[
\quad H(N_C, n_S, N) \triangleright (\langle M \rangle.P)
\]

\[
s[TLS, C, S].((?m).P) \triangleq (C, S, ?n_C).
\]
\[
\quad (\nu N_S)\langle S, C, N_S, S_{K_{CA}^-}^-(S, K_S^+) \rangle
\]
\[
\quad (C, S, P_{K_{CA}^-}^-(?n)).
\]
\[
\quad H(n_C, N_S, n) \triangleright ((?m).P)
\]
Concrete Specification of Client-Bank Example

The overall system takes the form

\[ \text{System} \triangleq (\nu \pm K_{CA})(\nu \pm K_{Bank}) \ (\text{Client} \ | \ Bank) \]

where the client and bank are now given as follows:

\[ \text{Client} \triangleq (\nu \ TS)(\nu \ N_C)\langle \text{Client, Bank, } N_C \rangle. \]
\[ (Bank, Client, ?n_b, S_{K_{CA}^+}(Bank, K_{Bank}^+)) \]
\[ (\nu \ N)\langle \text{Client, Bank, } P_{K_{Bank}^+}(N) \rangle. \]
\[ H(N_C, n_b, N) \triangleright (\langle \text{Client, Bank, TS} \rangle.(Bank, Client, TS, ?bal).0 ) \]

\[ \text{Bank} \triangleq (\text{Client, Bank, } ?n_c). \]
\[ (\nu \ N_B)\langle \text{Bank, Client, } N_B, S_{K_{CA}^-}(Bank, K_{Bank}^-) \rangle \]
\[ (\text{Client, Bank, } P_{K_{Bank}^-}(?n)). \]
\[ H(n_c, N_B, n) \triangleright (\text{Client, Bank, } ?ts).(\nu \ Bal) \langle \text{Bank, Client, ts, Bal} \rangle.0 \]
Establish a Bridge – An Unfolding Function

- systematically transform processes outside of the concrete level into the concrete level

- 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

\[
\bar{s}[p_{i_1}; \ldots; p_{i_k}].P \triangleq \bar{s}[p_{i_k}].(\ldots(\bar{s}[p_{i_1}].P))
\]

\[
s[p_{i_1}; \ldots; p_{i_k}].P \triangleq s[p_{i_k}].(\ldots(s[p_{i_1}].P))
\]
Credit Request Case Study
Credit Request Case Study

Protocol Stacks for Services
Credit Request Case Study

Diagram showing the interaction between a customer, a bank, and validation services for corporates and enterprises. The process includes sending a Credit Request (Credit_Req) from the customer to the bank, which then communicates with a SOAP-Mediator that forwards the request to validation services for corporates and enterprises.
Abstract Level Modelling

\[ \text{System} \triangleq \text{Client} \mid \text{Bank} \mid \text{Ser}_E \]

\[ \text{Client} \triangleq ! (\nu \text{Id})(\nu \text{Bta}) \]
\[ \text{credit}_\text{req}[ \ ].(\langle \text{Id}, \text{Bta}\rangle.(\text{Id}, \text{Bta}, ?x_{res}). \uparrow \langle x_{res}\rangle.0) \]

\[ \text{Bank} \triangleq ! \text{credit}_\text{req}[ \ ].(?(y_{id}, y_{bta}). \]
\[ \text{validate}[ \ ].\langle y_{bta}\rangle.(?y_{res}). \uparrow \langle y_{id}, y_{bta}, y_{res}\rangle.0) \]

\[ \text{Ser}_E \triangleq ! \text{validate}[ \ ].(?(z_{bta}).\langle \text{isValid}(z_{bta})\rangle.0) \]

\( \text{Id} \) : Identification number of the Request

\( \text{Bta} \) : Balance Total Assets
Go one step further and consider the various protocols needed to secure the communication.

- **TLS (Transport Layer Security):** a key establishment protocol.

- **WS-Security (Web Service Security):** a communication protocol suite providing security to Web services, while guaranteeing point-to-point integrity and authenticity.

- **SOAP (Simple Object Access Protocol):** a protocol specification for exchanging structured information in computer networks.
**Protocol Layers**

- **A** is sending a message $M$ to **B**
  - **A** may insert its certificate in the message in case it is unknown to **B**, to allow **B** to reply.

- **WS-Security** signs the encrypted messages to ensure integrity and confidentiality

- **SOAP** gives a standardised format for the messages and adds the message header to allow SOAP-routing
Abstract Specification with Protocol Stack Plug-In

\[\text{Client} \triangleq ! (\nu \, \text{Id})(\nu \, \text{Bta}) \text{credit\_req}[\text{TLS, Client, Bank}]. \]
\[\left( \langle \text{Id}, \text{Bta} \rangle . (\text{Id}, \text{Bta}, ?x_{\text{res}}) . \uparrow \langle x_{\text{res}} \rangle . 0 \right)\]

\[\text{Bank} \triangleq ! \text{credit\_req}[\text{TLS, Client, Bank}]. (?y_{\text{id}}, ?y_{\text{bta}}). \]
\[\text{validate}[\text{WS, Bank, Ser}_{E}; \text{SOAP, Bank, SM, Ser}_{E}].\]
\[\left( \langle y_{\text{bta}} \rangle . (?y_{\text{res}}) . \uparrow \langle y_{\text{id}}, y_{\text{bta}}, y_{\text{res}} \rangle . 0 \right)\]

\[\text{Ser}_{E} \triangleq ! \text{validate}[\text{WS, Bank, Ser}_{E}; \text{SOAP, Bank, SM, Ser}_{E}].\]
\[\left( (?z_{\text{bta}}) . \langle \text{isValid}(z_{\text{bta}}) \rangle . 0 \right)\]

\[\text{SM} \triangleq ! (?A, \text{SM}, A, ?B, ?M) . \langle \text{SM}, B, A, B, M \rangle\]
Protocol Definitions

- **WS**: Sender sends out message $M$ together with his certificate. 
  $A \rightarrow B : M, S_{K_{CA}}^{-}(A, K_A^+)$. WS encrypts and signs the message.
  \[
  \bar{s}[WS, A, B].(\langle M \rangle.P) \triangleq \langle S_{K_{CA}}^{-}(P_{K_{B}^+}(M, S_{K_{CA}}^{-}(A, K_A^+))), \bar{s}[WS, A, B].P \rangle.
  \]
  \[
  \bar{s}[WS, A, B].((?m).P) \triangleq \langle S_{K_{A}^+}(P_{K_{B}^-}(?m, S_{K_{CA}^+}(A, ?k_a^+))), \bar{s}[WS, A, B].P \rangle.
  \]

- **SOAP**: appends additional fields to messages.
  \[
  \bar{s}[SOAP, A, SM, B].(\langle M \rangle.P) \triangleq \langle A, SM, A, B, M \rangle.
  \]
  \[
  \bar{s}[SOAP, A, SM, B].P
  \]
  \[
  s[SOAP, A, SM, B].((?m).P) \triangleq \langle SM, B, A, B, ?m \rangle.
  \]
  \[
  s[SOAP, A, SM, B].P
  \]
Establish a Bridge – An Unfolding Function

- systematically transform processes outside of the concrete level into the concrete level
- 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

\[
\bar{s}[p_1; \ldots; p_k].P \triangleq \bar{s}[p_k].(\ldots(\bar{s}[p_1].P))
\]

\[
s[p_1; \ldots; p_k].P \triangleq s[p_k].(\ldots(s[p_1].P))
\]
Unfolding Result

Client $\triangleq ! (\nu \text{Id})(\nu \text{Bta})$

$(\nu \text{NA})(A, B, \text{NA}).(B, A, ?x_n, S_{K^+_{CA}}(B, K_B^+))$

$(\nu \text{NPMS})(A, B, P_{K^+_{B}}(\text{NPMS})).$

$H(\text{NA}, x_n, \text{NPMS}) \triangleright (\langle \text{Id}, \text{Bta}\rangle.(\text{Id}, \text{Bta}, ?x_{res}).\langle x_{res}\rangle.0)$

Bank $\triangleq !$

$(A, B, ?y_n).(\nu \text{NB})(B, A, \text{NB}, S_{K^-_{CA}}(B, K_B^+))$

$(A, B, P_{K^-_{B}}(?y_{PMS})).$

$H(y_n, \text{NB}, y_{PMS}) \triangleright (?y_{id}, ?y_{bta}).$

$\langle \text{Bank}, \text{SM}, \text{Bank}, \text{Ser}_E, S_{K^+_{A}}(P_{K^+_{B}}(y_{bta}, S_{K^-_{CA}}(A, K_A^+))).$

$(SM, \text{Bank}, \text{Ser}_E, \text{Bank}, S_{K^+_{A}}(P_{K^-_{B}}(?y_{res}))).$

$\langle y_{id}, y_{bta}, y_{res}\rangle.0$

Ser$_E$ $\triangleq !$

$(SM, \text{Ser}_E, \text{Bank}, \text{Ser}_E, S_{K^+_{A}}(P_{K^-_{B}}(?z_{bta}, S_{K^+_{CA}}(A, ?k_a^+))).$

$(\langle \text{Ser}_E, \text{SM}, \text{Ser}_E, \text{Bank}, S_{K^-_{A}}(P_{K^+_{B}}(\text{isValid}(z_{bta}))).\langle z_{bta}\rangle.0)$

SM $\triangleq !$

$(?A, \text{SM}, A, ?B, ?M).\langle \text{SM}, B, A, B, M\rangle.0$
A static control flow analysis for concrete specifications in CaPiTo.

Focusing on authentication – adding annotations about the origin and destination of encrypted messages

\[ e \triangleright \langle V_1, V_2 \rangle^1[dest \{l_2\}].0 \]
\[ e \triangleright (V_1, x)^2[orig \{l_1\}].0 \]

Similar annotations are made to asymmetric encryption, signature, decryption and signature validation.
The Analysis

The analysis collects information into the analysis components:

- $\rho : \mathcal{P}(\text{Var} \times \text{Val})$ records for each variable the set of names it may be bound to.

- $\kappa : \mathcal{P}(\text{Val}^*)$ records all the tuples that has been sent to a server.

- $\psi : \mathcal{P}(\text{Lab} \times \text{Lab})$ 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 \( \vartheta \) 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 \( \rho, \kappa \) and annotation violations \( \psi \).
Some Analysis Rules

- Analysis Rules of Expressions

\[ \rho \models n : \vartheta \quad \text{iff} \quad n \in \vartheta \]
\[ \rho \models x : \vartheta \quad \text{iff} \quad \rho(x) \subseteq \vartheta \]
\[ \rho \models P_{v_0^+}(v_1, \ldots, v_k)[\text{dest } \mathcal{L}] : \vartheta \]
\[ \text{iff} \quad \bigwedge_{i=0}^{k} \rho \models v_i : \vartheta_i \wedge \\
\forall w_0, w_1, \ldots, w_k : \bigwedge_{i=0}^{k} w_i \in \vartheta_i \Rightarrow \\
P_{w_0^+}(w_1, \ldots, w_k)[\text{dest } \mathcal{L}] \in \vartheta \]

- Analysis Rules of Processes

\[ \rho, \kappa \models \langle e_1, \ldots, e_k \rangle.P : \psi \]
\[ \text{iff} \quad \bigwedge_{i=1}^{k} \rho \models e_i : \vartheta_i \wedge \\
\forall w_1, \ldots, w_k : \bigwedge_{i=1}^{k} w_i \in \vartheta_i \Rightarrow \langle w_1, \ldots, w_k \rangle \in \kappa \wedge \\
\rho, \kappa \models P : \psi \]
The results are computed in low polynomial time in the size of the system. In practice, less than two seconds.
Analyse the Cast Study

The overall scenario of the case study takes the form

\[ ((\nu \pm K_{VS})(\nu \pm K_{SerE})(\nu \pm K_{SerC}) \mid_{i=1}^{n} Client_i \mid VS \mid SM \mid Ser_E \mid Ser_C) \mid Attacker \]

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

The analysis gives an empty $\psi$ component

- no authentication annotations are violated

Some entries may be of interest:

\[
\begin{align*}
\rho(xr_1) &= \{ isValid(Bta_1) \} \\
\rho(xr_2) &= \{ isValid(Bta_2) \}
\end{align*}
\]

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

- The CaPiTo approach to model service-oriented systems at different levels of abstraction
  - the abstract level allows us to concentrate on the system itself
  - the concrete level allows us to take the underlying protocols into consideration
  - a transformation function

- A static analysis to track the run-time behaviour and check 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 partners and industrial partners