Setting the Standard for Automation™

ISA France

Generation of Applicative Attacks Scenarios Against Industrial Systems

Maxime Puys, Marie-Laure Potet, Jean-Louis Roch
VERIMAG, Univ. Grenoble Alpes

En partenariat avec

WATERFALL Stronger Than Firewalls

G-SCOP

WATERFALL Stronger Than Firewalls

HIMA SMART SAFETY.

sentryo

Verimag

UNIVERSITÉ Grenoble Alpes

Grenoble INP

Nouvelles vulnérabilités, nouvelles attaques, nouvelles solutions
Grenoble –30 et 31 janvier 2018

Industrial Systems (ICS) Composition 1/2

SCADA

PLC

Process

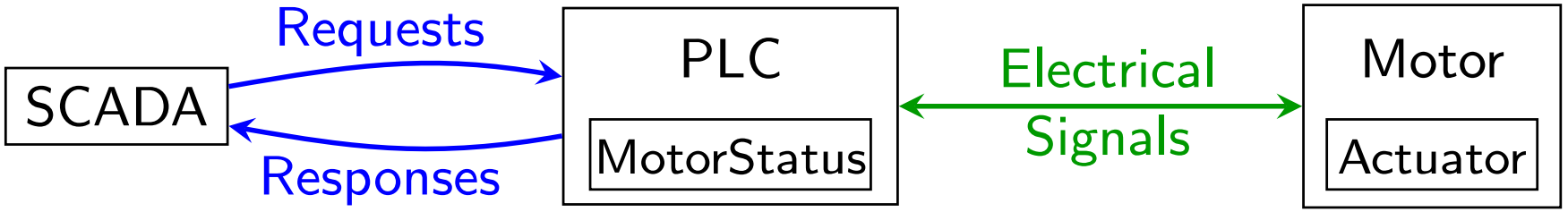
SCADA: Supervisory Control And Data Acquisition, controls and monitors the process.

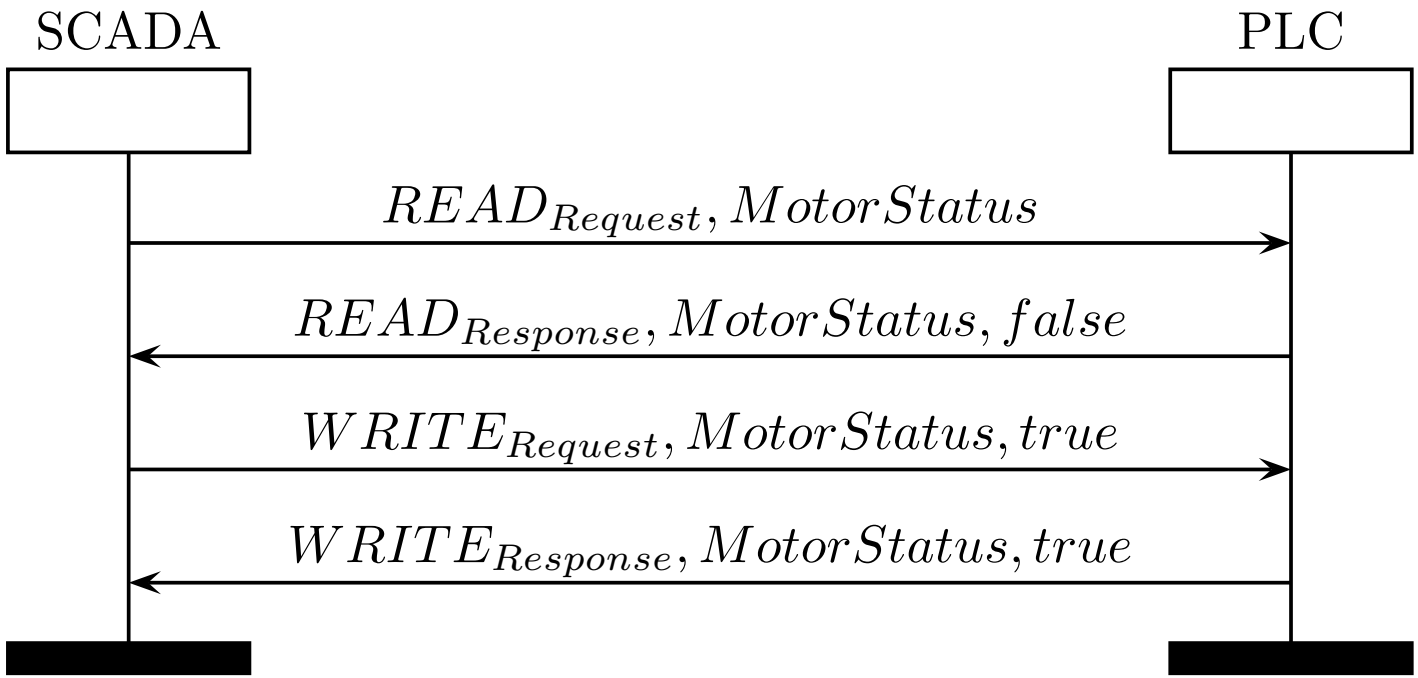
PLC: Programmable Logic Controller, interprets SCADA orders for the process.

Process: Actual industrial process managed by the system.

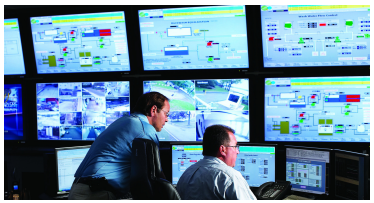
Industrial Systems (ICS) Composition 2/2

- Variables on PLC synchronized with process.
- Protocols used are specific (e.g.: MODBUS, OPC-UA).



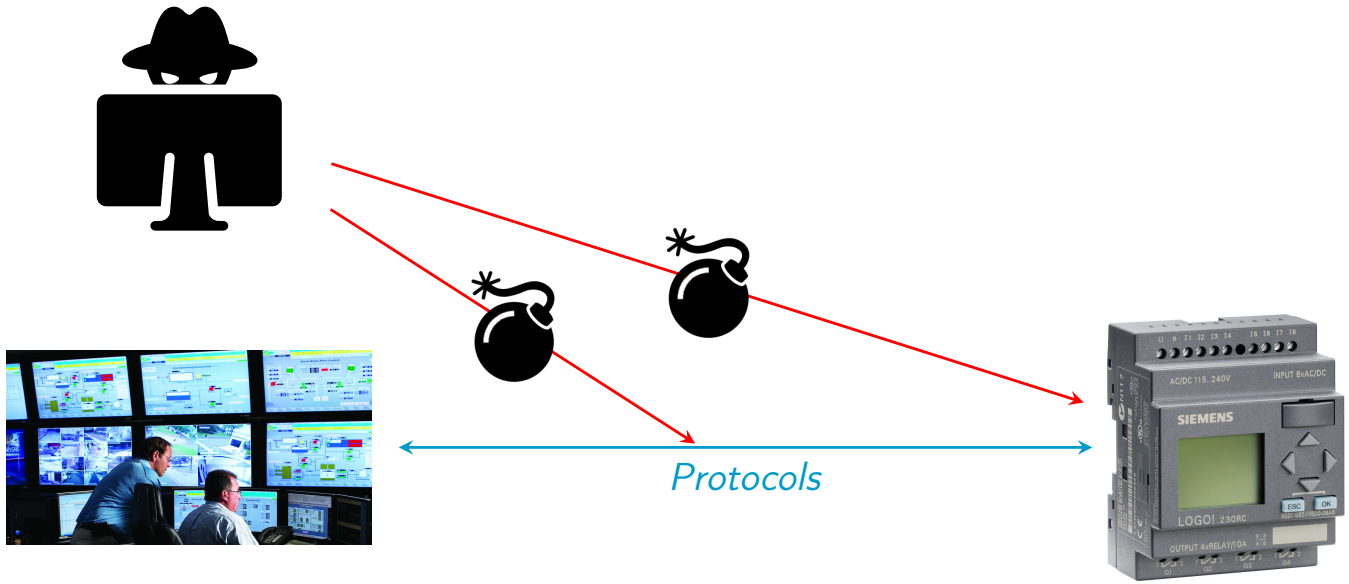


Overview of the Thesis

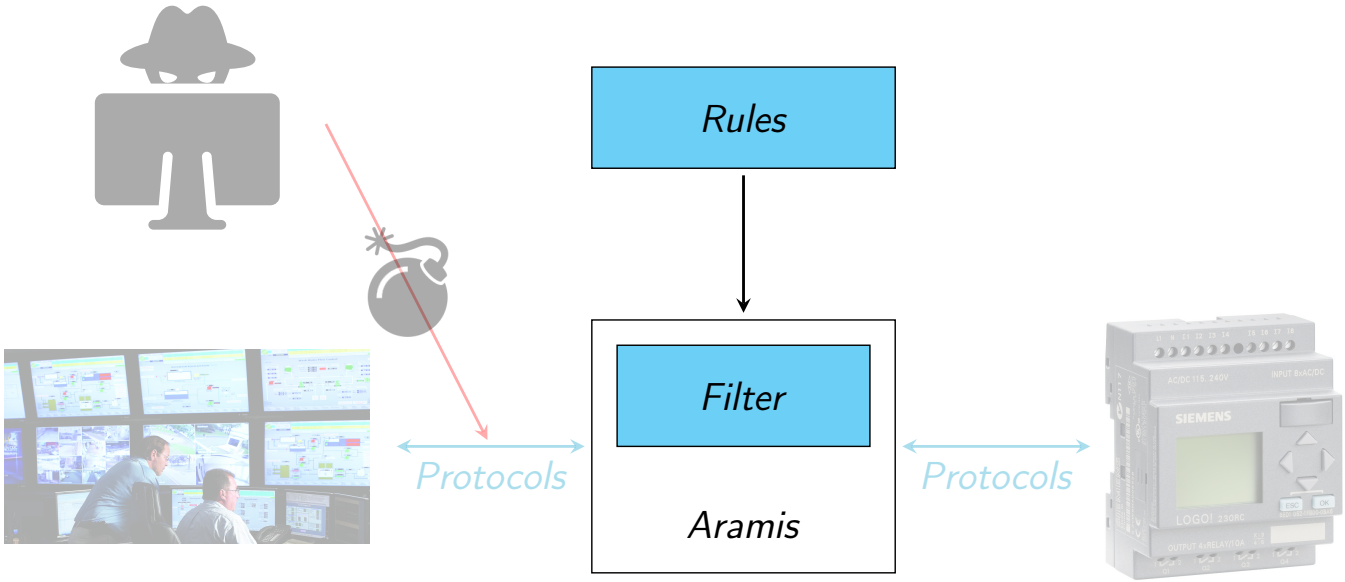


Protocols

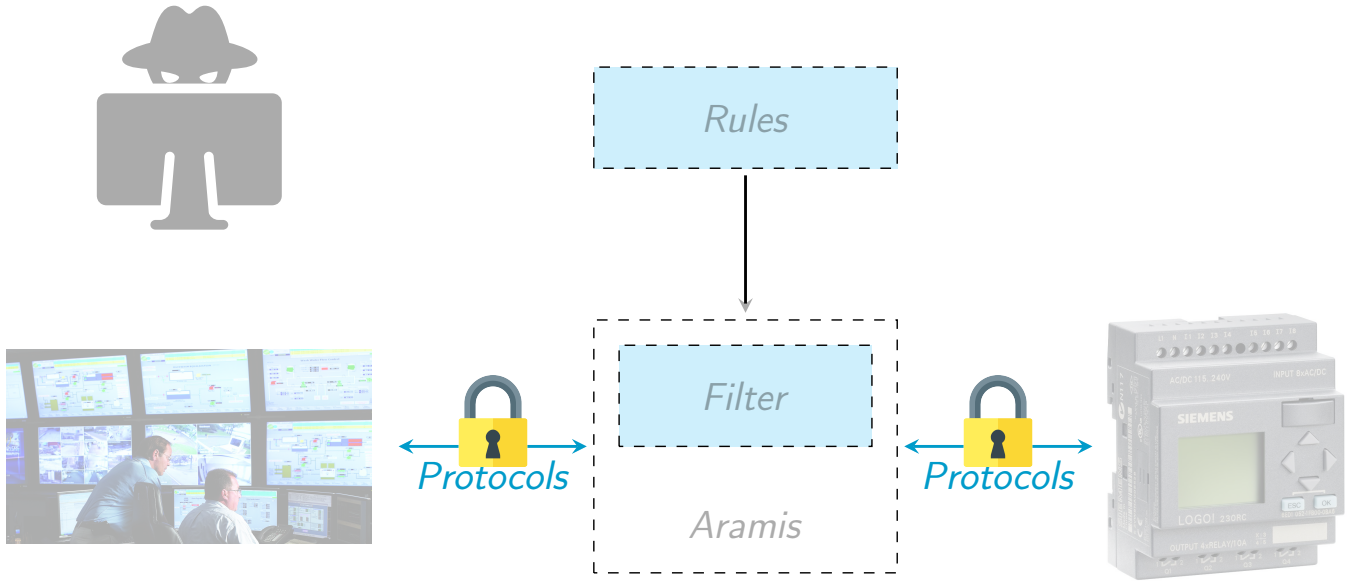
Overview of the Thesis



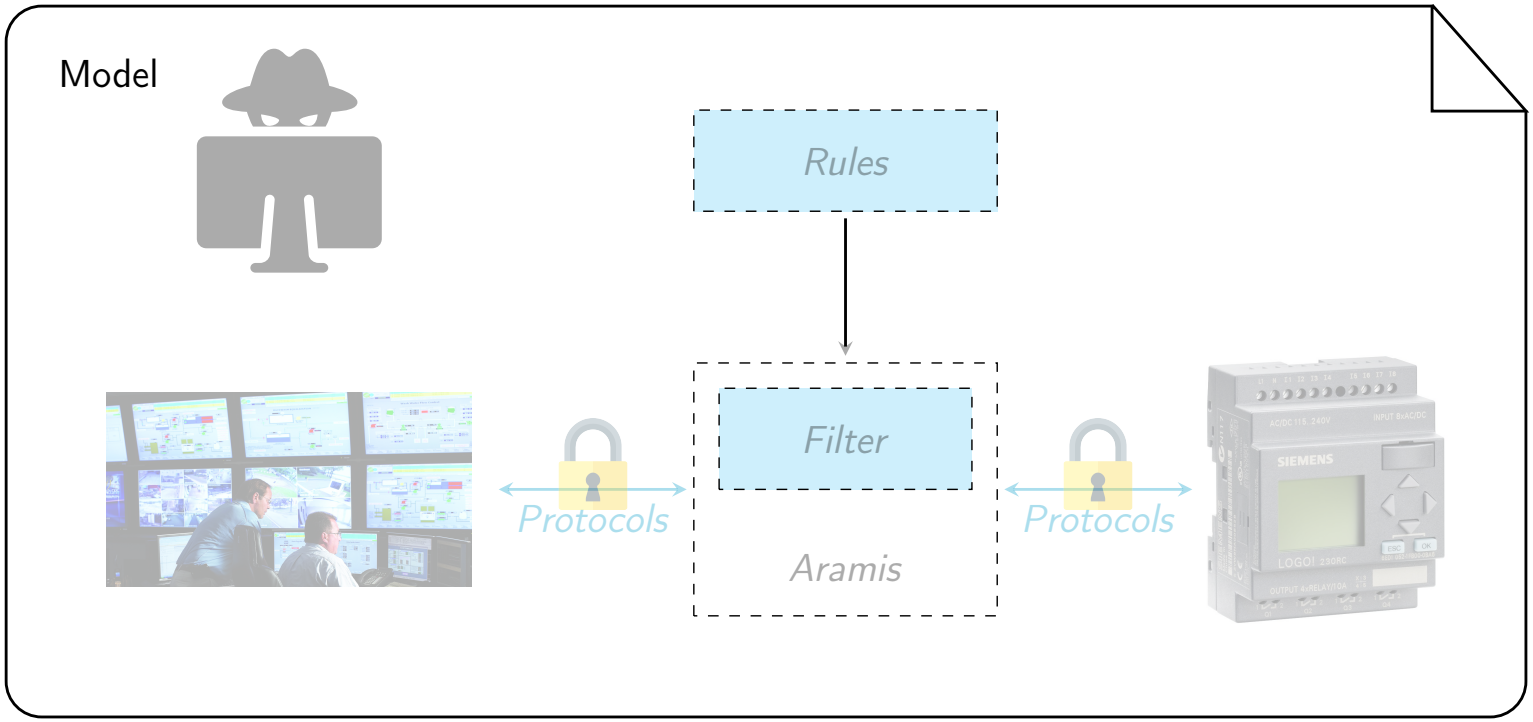
Overview of the Thesis: 1 – Applicative Filtering



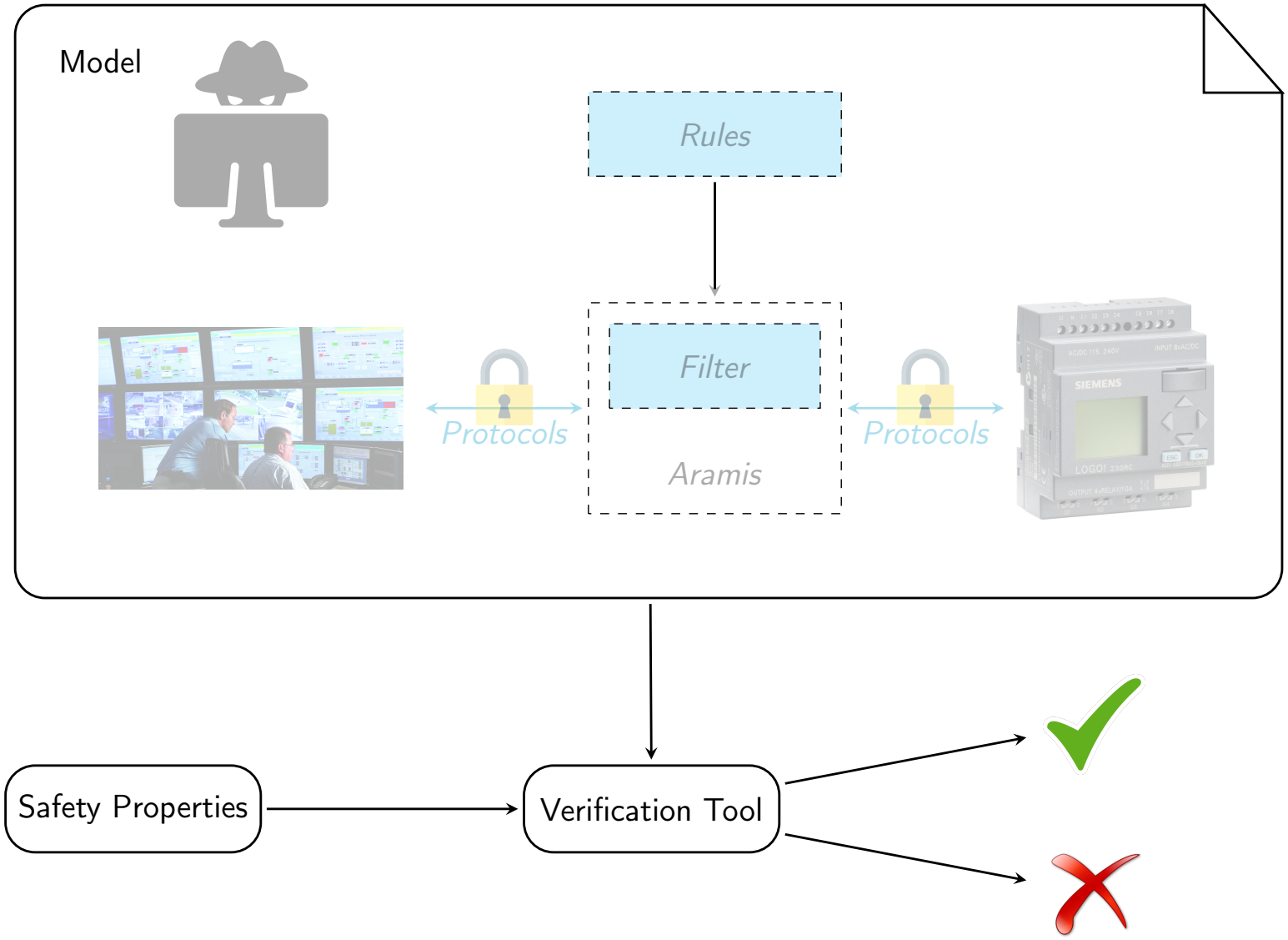
Overview of the Thesis: 2 – Protocol Verification



Overview of the Thesis: 3 – Attack Scenarios Generation

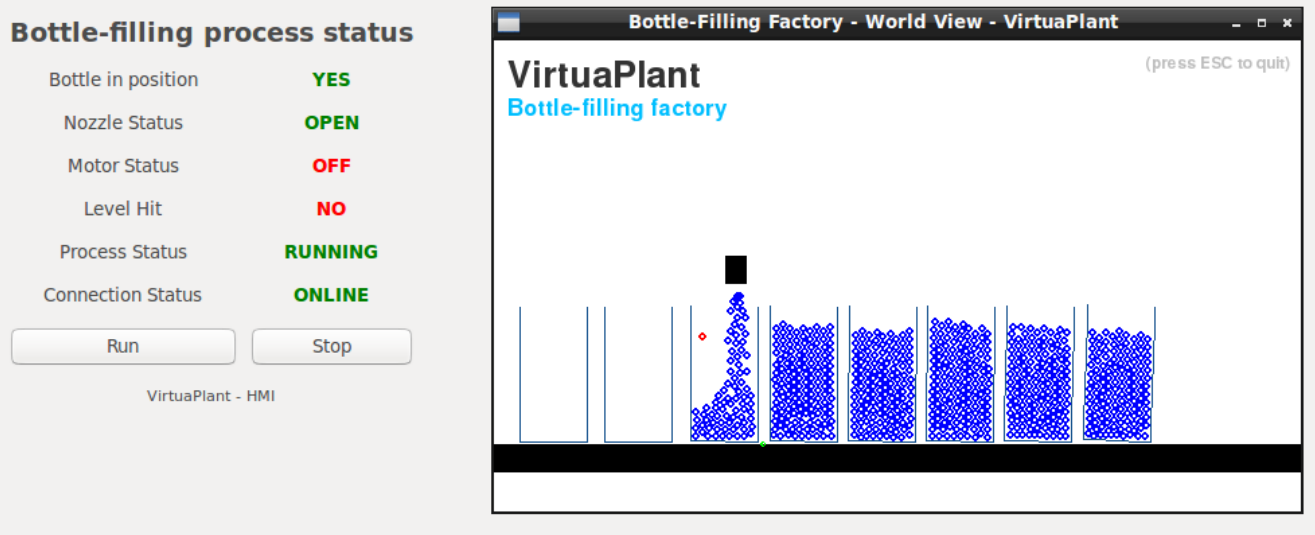


Overview of the Thesis: 3 – Attack Scenarios Generation



Case Study: Bottle-filling Factory

- Process simulator: `https://github.com/jseidl/virtuaplant`



Variables:

- Conveyor belt
- Nozzle
- Position captor
- Level captor
- On/Off Switch

Properties:

- Nozzle only opens when a bottle is detected.
- Conveyor belt only starts when the bottle is full.
- Nozzle only opens when conveyor belt is stopped.

Idea & Contributions

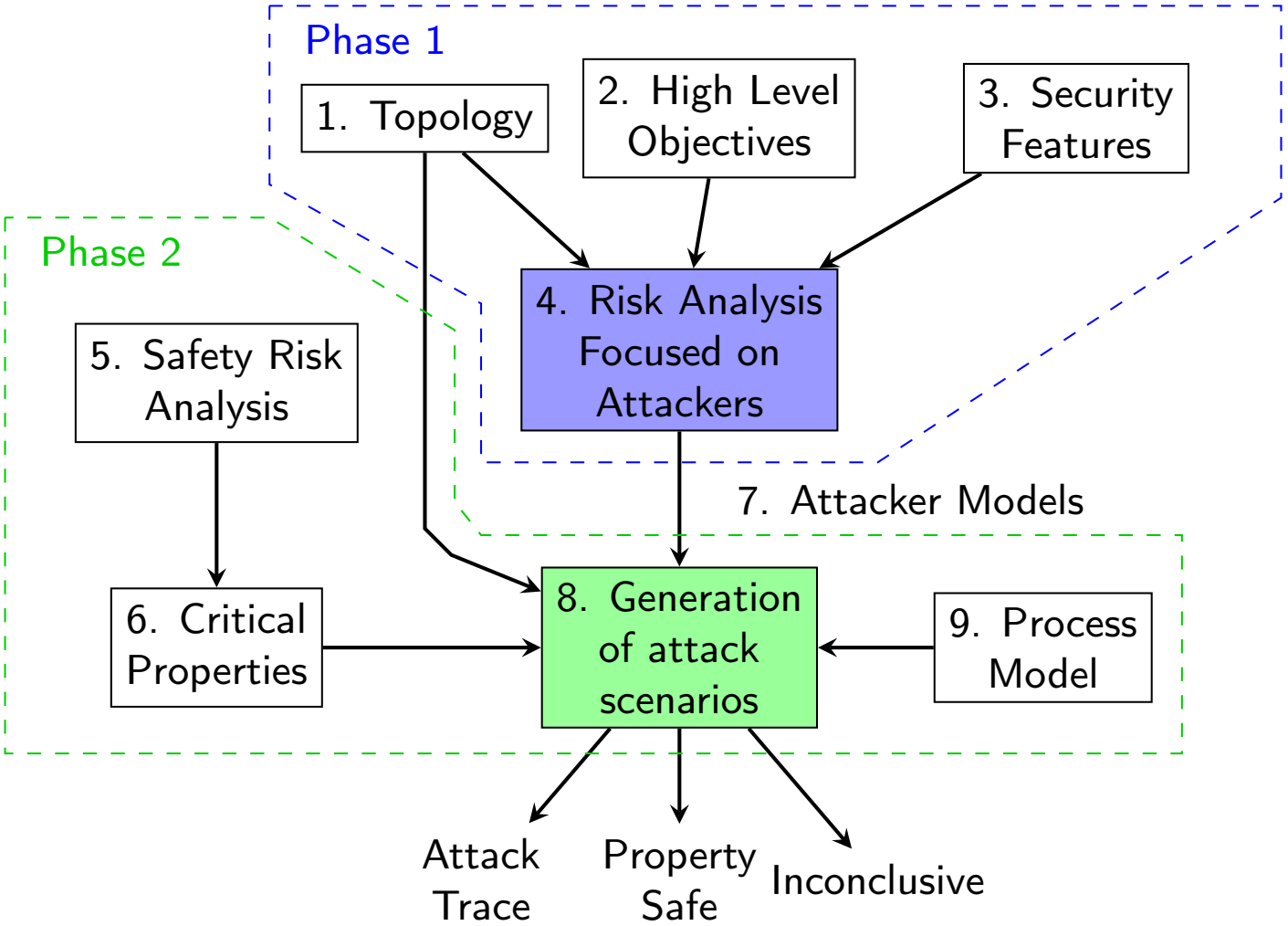
- A^{2}SPICS: Find **applicative attacks** on industrial systems:
  - Considering an attacker already in the system;
  - What possible actions on the industrial process.

Generic verification tools vs. Protocol verification tools

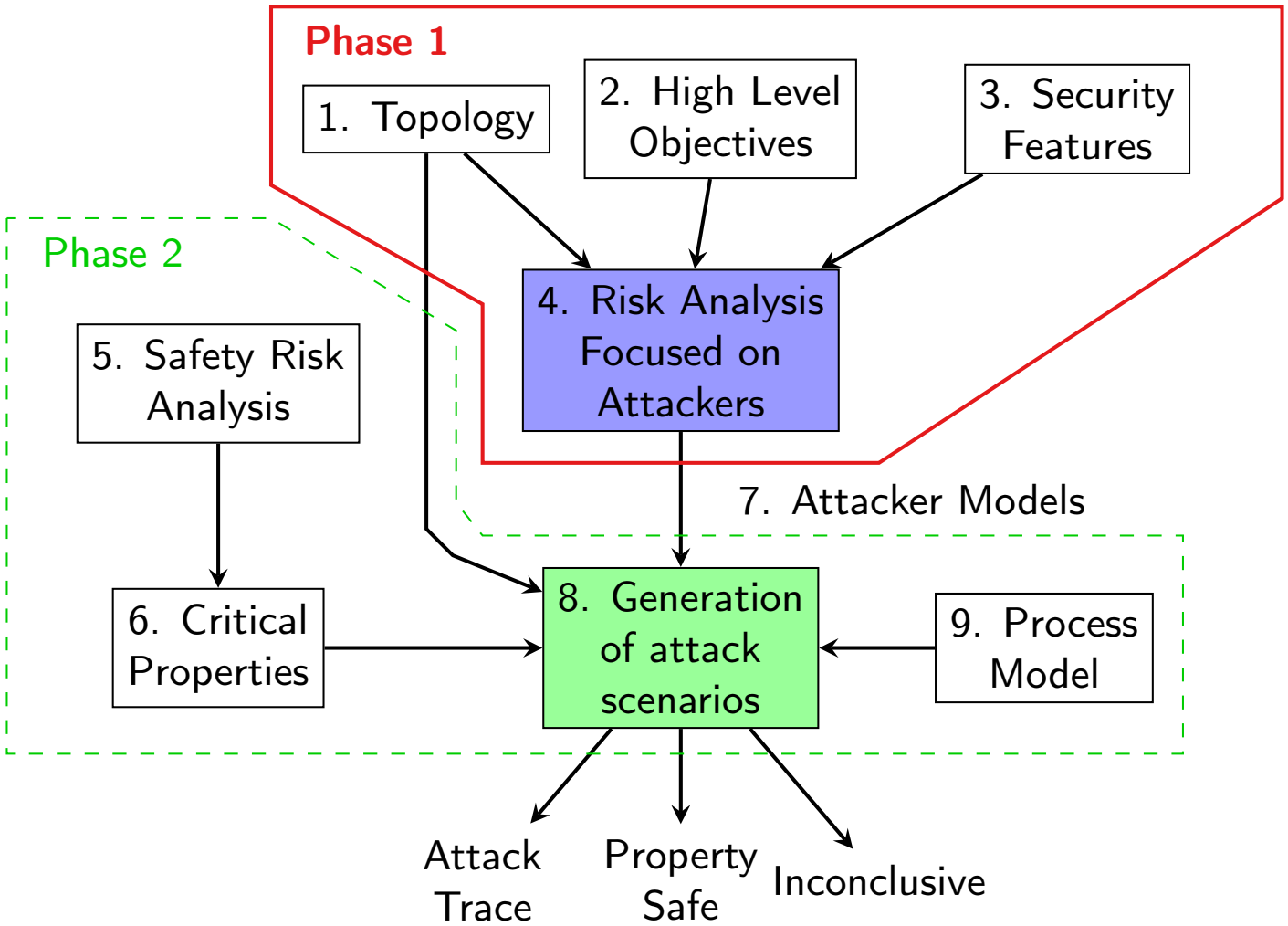
- Generic tools: model-checkers, smt-solvers, etc.
- Protocol verification tools: embed attacker logic.
- Trade-off: tool optimized for verification with attackers vs. states.

[FPS'17] M. Puys, M-L. Potet, and A. Khaled., 2016.

The A^{2}SPICS Approach

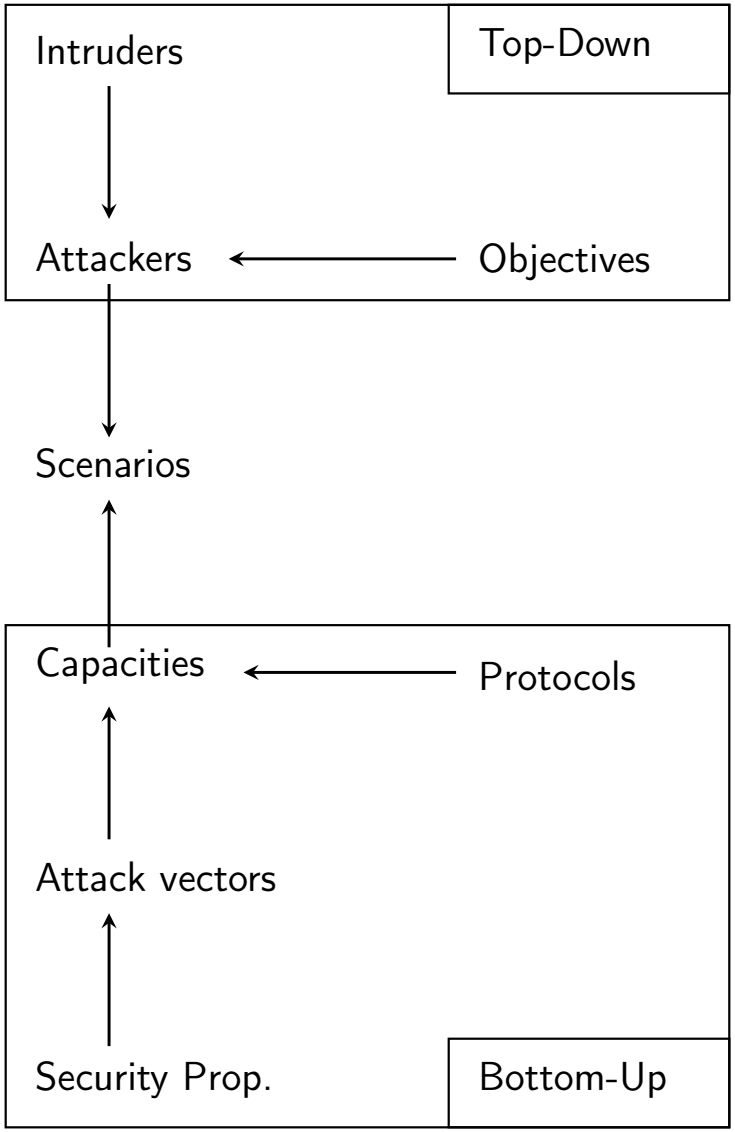


The A^{2}SPICS Approach



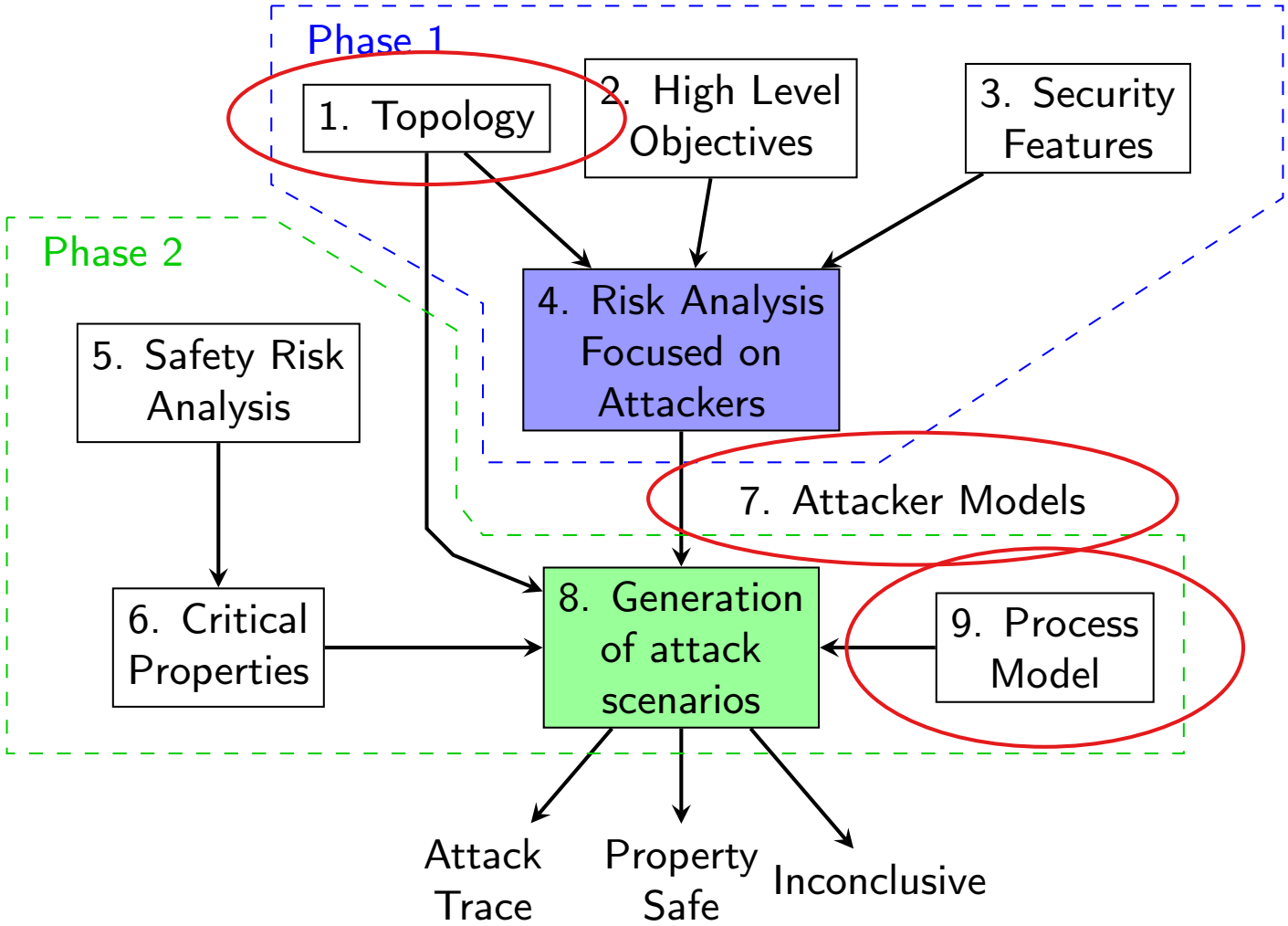
Phase 1: Attacker Models

- Risk analysis focused on attackers.
- Based on:
  - **Topology** of the system;
  - Attacker **objectives**;
  - **Security features** of protocols.
- Objectives are security vuln., e.g.:
  - Modify a message;
  - Circumvent authentication.
- Yields **attacker models** in terms of:
  - Position in the topology;
  - Capacities (actions and deduction).



[AFADL'16] M. Puys, M-L. Potet, and J-L. Roch., 2016.

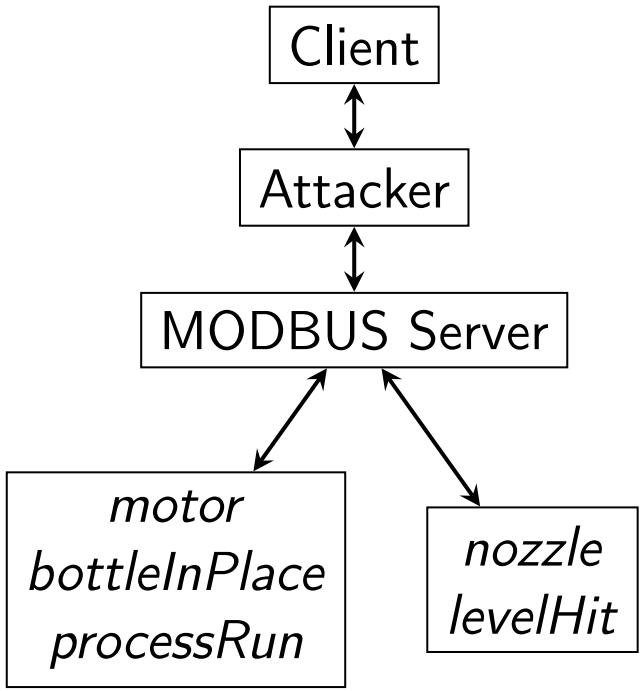
The A^{2}SPICS Approach

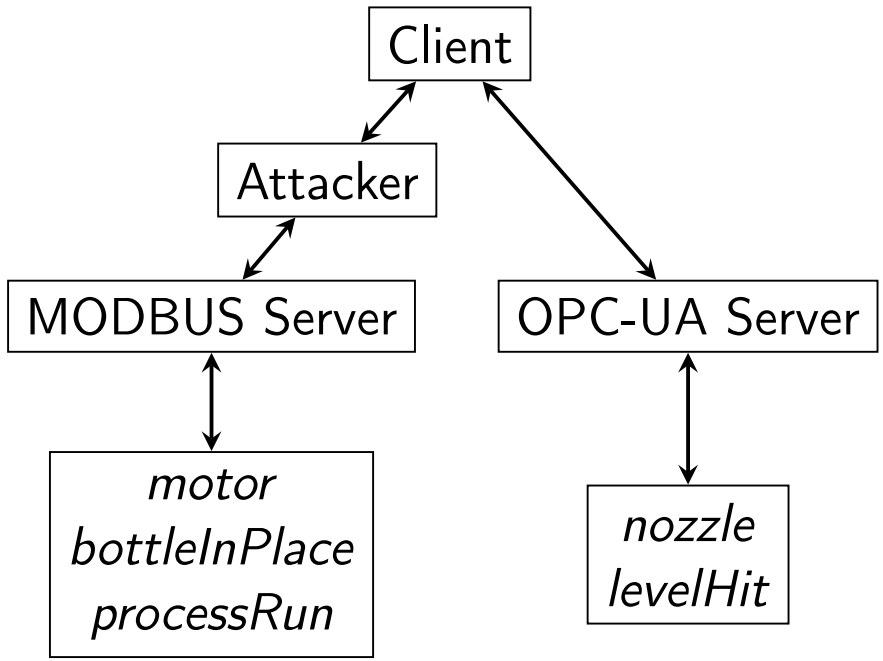


Topologies

Network topology of the system (expressed in CSP, π-calculus, etc):

- **Communication channels** between components;
- **Position** of attackers.



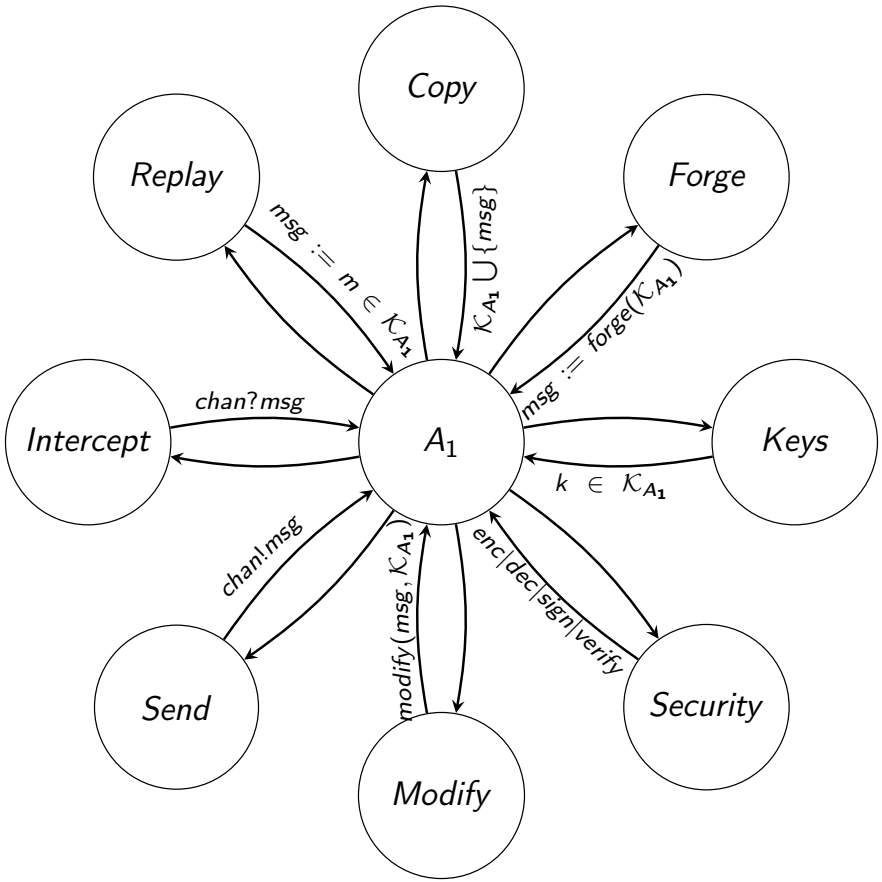


Attackers 1/2

Characterized by:

- **Position** in the topology:
  - On a channel (Man-In-The-Middle);
  - On a corrupted component (virus, malicious operator, etc).
- Capacities:
  - Possible **actions on messages** (intercept, modify, replay, etc);
  - **Deduction system** (deduce new information from knowledge, e.g.: encrypt/decrypt).
- Initial knowledge:
  - Other components;
  - Process behavior;
  - Cryptographic keys, etc.

Attackers 2/2

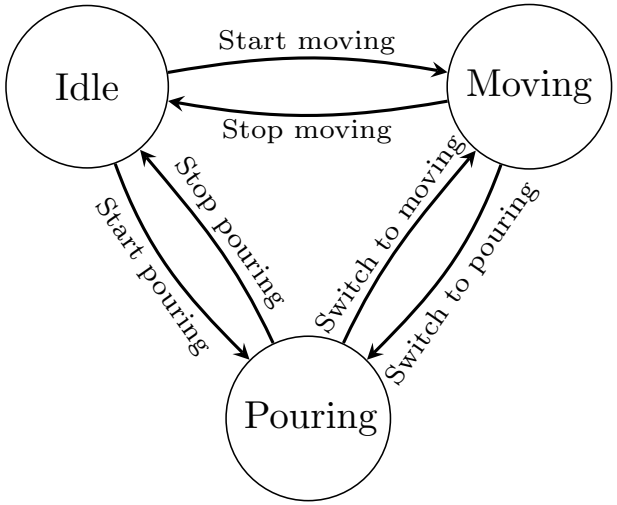


Four attackers:

- A_1 = close to Dolev-Yao;
- As instance, A_2, A_3 and A_4 are subsets of A_1.

Attacker	Modify	Forge	Replay
A_1	✓	✓	✓
A_2	✓	✗	✗
A_3	✗	✓	✗
A_4	✗	✗	✓

Behaviors and Safety Properties



Automaton of the behavior of the process

Current State	Next State	Guard	Actions
Idle	Moving	$processRun = true \wedge bottleInPlace = false$	$motor := true$
Idle	Pouring	$processRun = true \wedge bottleInPlace = true$	$nozzle := true$
Moving	Pouring	$bottleInPlace = true$	$motor := false \wedge nozzle := true$
Pouring	Moving	$levelHit = true$	$motor := true \wedge nozzle := false$
Moving	Idle	$processRun = false$	$motor := false \wedge nozzle := false$
Pouring	Idle	$processRun = false$	$motor := false \wedge nozzle := false$

Transitions Details

Properties: CTL formula:

- Φ_1: At all time and on each path, *nozzle* is never *true* if *bottleInPlace* is *false*). $A\Box \neg(nozzle = true \text{ and } bottleInPlace = false)$
- Φ_2: $A\Box \neg(motor = true \text{ and } levelHit = false)$
- Φ_3: $A\Box \neg(nozzle = true \text{ and } motor = true)$

Instrumentation using Different Tools

Implementation of A^{2}SPICS using 3 different tools:

- UPPAAL:
  - Model-checker, 1994.
  - Mainly designed for timed automata.
    $\Rightarrow$ **Safety** oriented verification tool.

- ProVerif:
  - Protocol verification tool, 2001.
    $\Rightarrow$ **Security** oriented verification tool.
  - Relying on π-calculus and Horn clauses.

- Tamarin:
  - Protocol verification tool, 2012.
    $\Rightarrow$ **Security** oriented verification tool.
  - Relying on Maude-NPA rewriting tool.

Results on the case study

All attackers on all properties (checked using UPPAAL):

- ✓ = attack found;
- ✗ = no attack found;
- $\mathcal{O}$ = inconclusive (here, out of memory).

Topologies	Properties	A_1	A_2	A_3	A_4
T_1	Φ_1	✓	✓	✓	✗
	Φ_2	✓	✓	✓	✗
	Φ_3	✓	✓	✓	✗
T_2	Φ_1	$\mathcal{O}$	$\mathcal{O}$	✗	✗
	Φ_2	✓	✓	✓	✗
	Φ_3	✓	✓	✓	✗

Limitations and Difficulties

For all three tools:

- Discretized time and process state (e.g.: tank either empty or full).
- Model limited to variables values.

UPPAAL: Attacker behavior is infinite

- Number of actions per attack is bounded (configurable, classical limitation of model-checking).

ProVerif: Very tedious state modeling

- Requires resilient channels, value enumeration, etc.

Tamarin: Impossible state modeling

- Backward search loops if behaviors has cycles.

Related Works

- Survey on assessment of security in industrial system ([PCB13, KPCBH15, CBB+15]).
- Comparison criteria from [KPCBH15, CBB+15]:

Ref.	Type	Focus	Process model	Probabilistic	Automated
[BFM04]	Model	A	No	No	No
[MBFB06]	Model	A	No	Yes (E)	No
[PGR08]	Model	A	No	Yes (E,H)	No
[TML10]	Model	A	No	Yes (H)	Yes
[CAL+11]	Formula	N/A	Yes	Yes (N/C)	Yes
[KBL15]	Model	A	No	Yes (E)	Yes
[RT17]	Model	A,G	Yes	No	Yes
A^{2}SPICS	Model	A,G	Yes	No	Yes

- [RT17] rely on CI-Atse (protocol verification tool):
  - Dolev-Yao intruder $\Rightarrow$ less precise control on attacker capacities.
- A^{2}SPICS aims at modeling **attackers resulting on risk analysis**.

Related Works

- Survey on assessment of security in industrial system ([PCB13, KPCBH15, CBB+15]).
- Comparison criteria from [KPCBH15, CBB+15]:

Ref.	Type	Focus	Process model	Probabilistic	Automated
[BFM04]	**Model**	**A**	No	No	No
[MBFB06]	**Model**	**A**	No	Yes (E)	No
[PGR08]	**Model**	**A**	No	Yes (E,H)	No
[TML10]	**Model**	**A**	No	Yes (H)	Yes
[CAL+11]	Formula	N/A	Yes	Yes (N/C)	Yes
[KBL15]	**Model**	**A**	No	Yes (E)	Yes
[RT17]	Model	A,G	Yes	No	Yes
A^{2}SPICS	Model	A,G	Yes	No	Yes

- [RT17] rely on CI-Atse (protocol verification tool):
  - Dolev-Yao intruder $\Rightarrow$ less precise control on attacker capacities.
- A^{2}SPICS aims at modeling **attackers resulting on risk analysis**.

Related Works

- Survey on assessment of security in industrial system ([PCB13, KPCBH15, CBB+15]).
- Comparison criteria from [KPCBH15, CBB+15]:

Ref.	Type	Focus	Process model	Probabilistic	Automated
[BFM04]	Model	A	No	No	No
[MBFB06]	Model	A	No	Yes (E)	No
[PGR08]	Model	A	No	Yes (E,H)	No
[TML10]	Model	A	No	Yes (H)	Yes
[CAL+11]	Formula	N/A	Yes	Yes (N/C)	Yes
[KBL15]	Model	A	No	Yes (E)	Yes
[RT17]	Model	A,G	Yes	No	Yes
A^{2}SPICS	Model	A,G	Yes	No	Yes

- [RT17] rely on CI-Atse (protocol verification tool):
  - Dolev-Yao intruder $\Rightarrow$ less precise control on attacker capacities.
- A^{2}SPICS aims at modeling **attackers resulting on risk analysis**.

Related Works

- Survey on assessment of security in industrial system ([PCB13, KPCBH15, CBB+15]).
- Comparison criteria from [KPCBH15, CBB+15]:

Ref.	Type	Focus	Process model	Probabilistic	Automated
[BFM04]	Model	A	No	No	No
[MBFB06]	Model	A	No	**Yes (E)**	No
[PGR08]	Model	A	No	**Yes (E,H)**	No
[TML10]	Model	A	No	**Yes (H)**	Yes
[CAL+11]	Formula	N/A	Yes	**Yes (N/C)**	Yes
[KBL15]	Model	A	No	**Yes (E)**	Yes
[RT17]	Model	A,G	Yes	No	Yes
A^{2}SPICS	Model	A,G	Yes	No	Yes

- [RT17] rely on CI-Atse (protocol verification tool):
  - Dolev-Yao intruder $\Rightarrow$ less precise control on attacker capacities.
- A^{2}SPICS aims at modeling **attackers resulting on risk analysis**.

Related Works

- Survey on assessment of security in industrial system ([PCB13, KPCBH15, CBB+15]).
- Comparison criteria from [KPCBH15, CBB+15]:

Ref.	Type	Focus	Process model	Probabilistic	Automated
[BFM04]	Model	A	No	No	No
[MBFB06]	Model	A	No	Yes (E)	No
[PGR08]	Model	A	No	Yes (E,H)	No
[TML10]	Model	A	No	Yes (H)	Yes
[CAL+11]	Formula	N/A	Yes	Yes (N/C)	Yes
[KBL15]	Model	A	No	Yes (E)	Yes
[RT17]	Model	A,G	Yes	No	Yes
A^{2}SPICS	Model	A,G	Yes	No	Yes

- [RT17] rely on CI-Atse (protocol verification tool):
  - Dolev-Yao intruder $\Rightarrow$ less precise control on attacker capacities.
- A^{2}SPICS aims at modeling **attackers resulting on risk analysis**.

Perspectives

- Assess example from [RT17] for a better comparison.
- Allow collusions between intruders.
- Consider resilience properties.

- Tentative of automation with ProVerif and Tamarin:
  - Apply formalisms of [RT17].

- Include ARAMIS device in model.

- Combine protocol and safety properties verification.

Conclusion

Thanks for your attention!

Maxime Puys

Maxime.Puys@univ-grenoble-alpes.fr

Shameless advertisement

Ph.D Defense on **Monday, February 5th at 10h30 at IMAG Building**.
Everyone is welcome.

References I


Eric J Byres, Matthew Franz, and Darrin Miller, *The use of attack trees in assessing vulnerabilities in scada systems*, Proceedings of the international infrastructure survivability workshop, 2004.


Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry, *Attacks against process control systems: risk assessment, detection, and response*, Proceedings of the 6th ACM symposium on information, computer and communications security, ACM, 2011, pp. 355–366.


Yulia Cherdantseva, Pete Burnap, Andrew Blyth, Peter Eden, Kevin Jones, Hugh Soulsby, and Kristan Stoddart, *A review of cyber security risk assessment methods for SCADA systems*, Computers & Security **56** (2015), 1 – 27.

References II


S Kriaa, M Bouissou, and Y Laarouchi, *A model based approach for SCADA safety and security joint modelling: S-Cube*, IET System Safety and Cyber Security, IET Digital Library, 2015.


Siwar Kriaa, Ludovic Pietre-Cambacedes, Marc Bouissou, and Yoran Halgand, *A survey of approaches combining safety and security for industrial control systems*, Reliability Engineering & System Safety **139** (2015), 156–178.


Miles A McQueen, Wayne F Boyer, Mark A Flynn, and George A Beitel, *Quantitative cyber risk reduction estimation methodology for a small scada control system*, System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on, vol. 9, IEEE, 2006, pp. 226–226.

References III


Ludovic Piètre-Cambacédès and Marc Bouissou, *Cross-fertilization between safety and security engineering*, Reliability Engineering & System Safety **110** (2013), 110–126.


Sandip C Patel, James H Graham, and Patricia AS Ralston, *Quantitatively assessing the vulnerability of critical information systems: A new method for evaluating security enhancements*, International Journal of Information Management **28** (2008), no. 6, 483–491.


Marco Rocchetto and Nils Ole Tippenhauer, *Towards formal security analysis of industrial control systems*, Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ACM, 2017, pp. 114–126.

References IV


Chee-Wooi Ten, Govindarasu Manimaran, and Chen-Ching Liu, *Cybersecurity for critical infrastructures: Attack and defense modeling*, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **40** (2010), no. 4, 853–865.


Theodore J Williams, *A reference model for computer integrated manufacturing (cim): A description from the viewpoint of industrial automation: Prepared by cim reference model committee international purdue workshop on industrial computer systems*, Instrument Society of America, 1991.

Industrial Systems are Ubiquitous

Electricity

Water Treatment

Chemistry

Industrial Systems are Ubiquitous

Electricity

Water Treatment

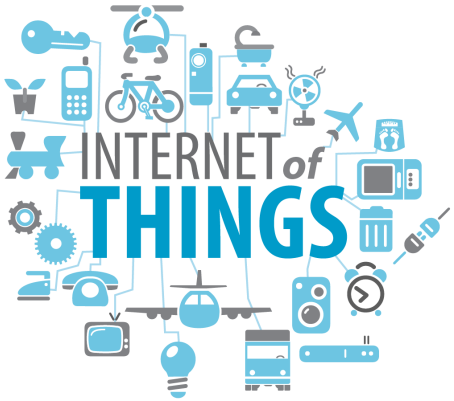
Chemistry

Food Production

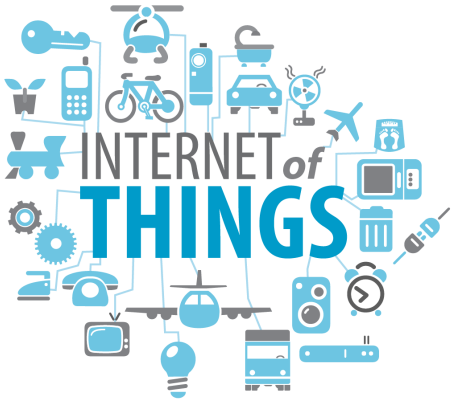
Transportation

Healthcare

Industrial Internet of Things



Industrial Internet of Things



Rio Tinto Mine, Australia

Oil Platform, North Sea

« Smart » Buildings

Autonomous Industrial Systems

Purdue Model

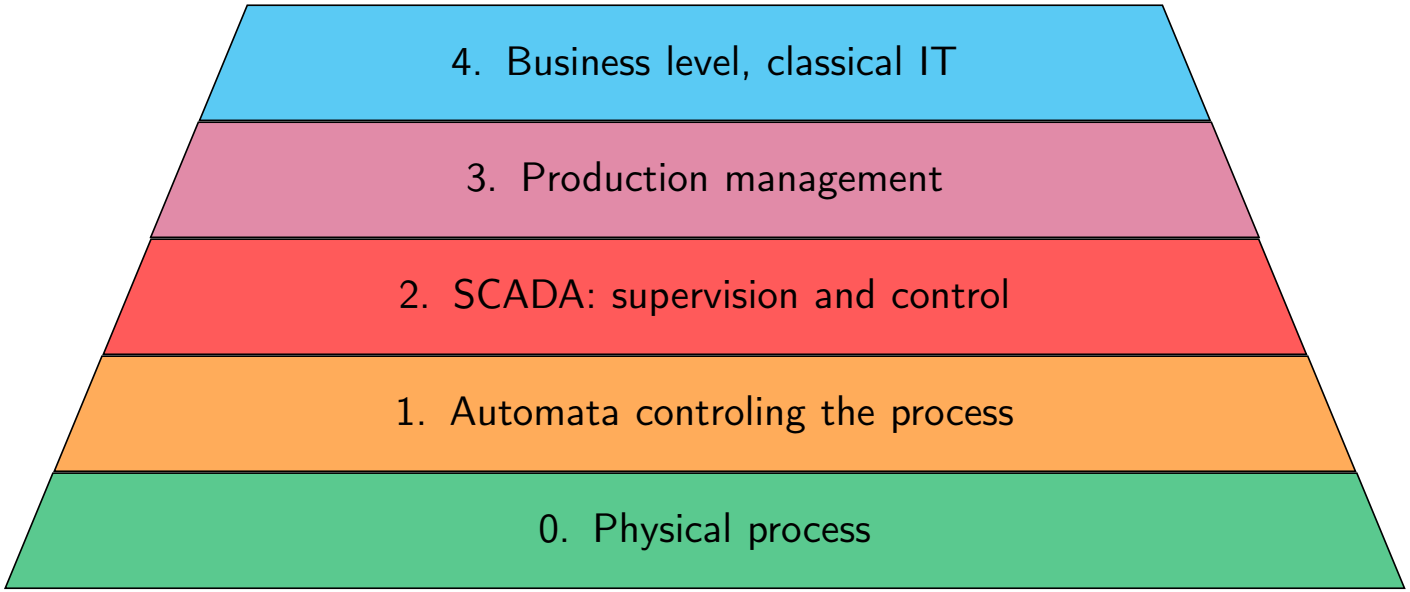


Figure : Purdue model [Wil91]

Norms and Guides on Industrial Systems Security

Generic

ISA-99/IEC-62443 (2007, 2013), ENISA (2011), ISO-27019 (2013), IEC-62541 (2015), etc.

Government Agency

CPNI (2008), BSI (2009), NIST (2011), ANSSI (2012), etc.

Domain Specific

Oil/Gaz (AGA, 2006), Electricity (IEC-62351, 2007]), Nuclear (IEC-62645, 2008), Air Traffic (CSFI, 2015), Railways (RSSB, 2016), etc.

Key Takeaways

$\Rightarrow$ Lots of documents, mainly released since 2006. Balanced partition between industry and governments, often in collaboration.

Properties to Ensure

For the process

Availability: System keeps running.

Integrity: Preservation of the coherence of a data over time.

Authenticity: An entity is who he/she pretends.

Non-repudiation: One cannot deny its actions.

Dependability: Domain specific properties.

For customer data

Confidentiality: Only authorized entities can access designated data.

Anonymity: Prevent linking a data with its owner.

Different Attackers



Different Attackers

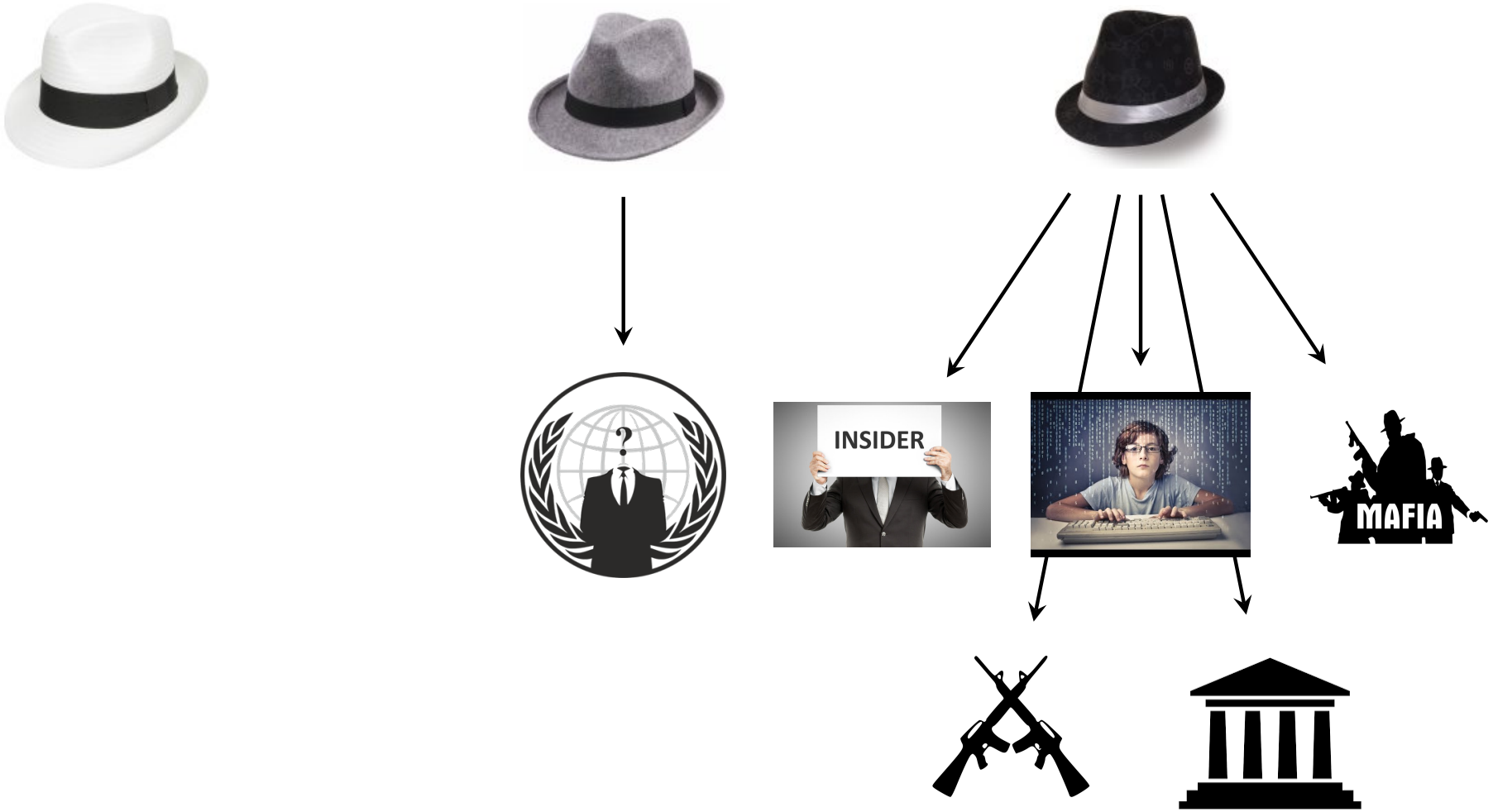
Different Attackers

Different Attackers

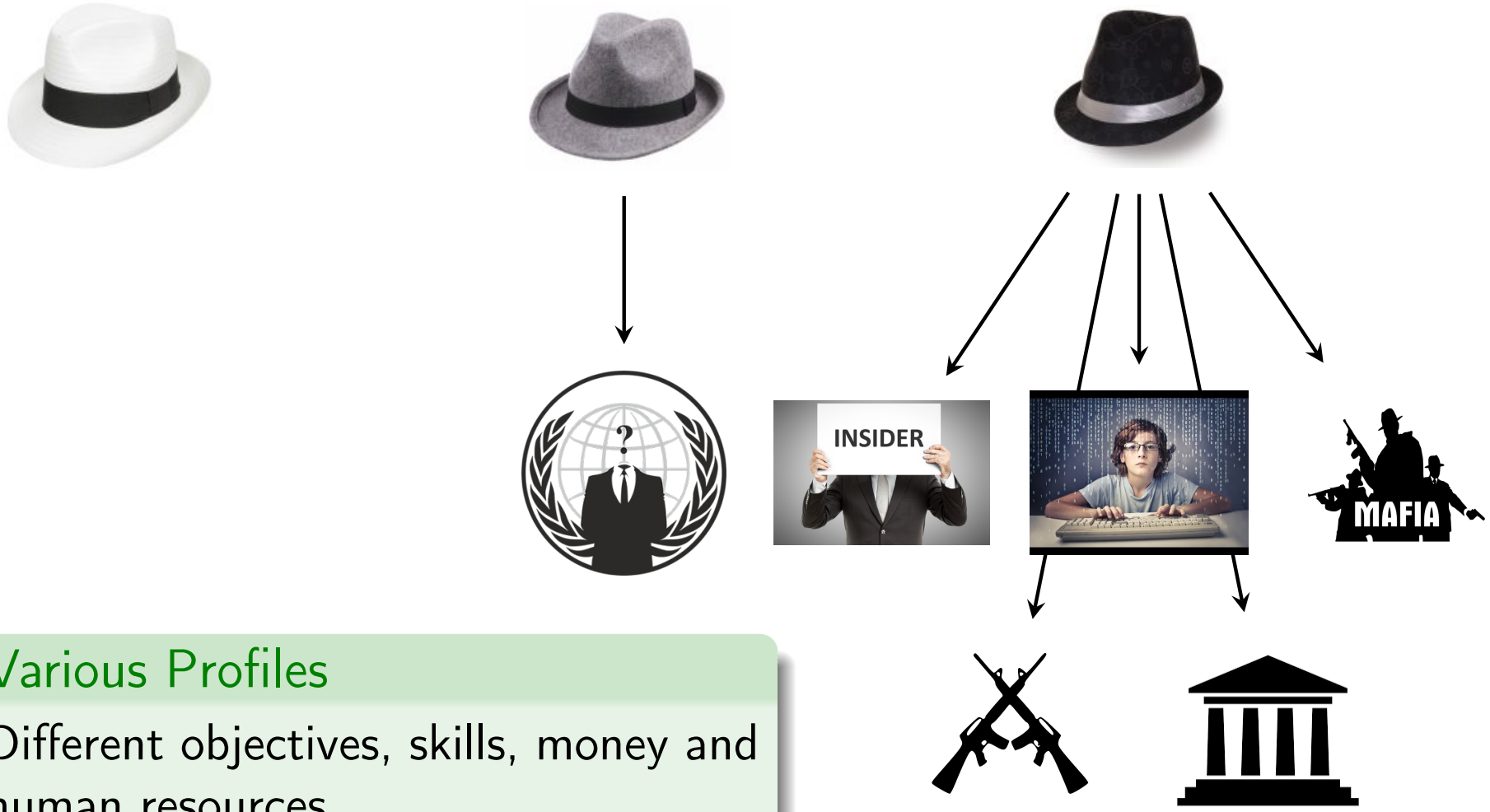
Different Attackers

Different Attackers

Different Attackers



Different Attackers



Various Profiles

Different objectives, skills, money and human resources.

Worst-Case Bandwidth

Both conditions and actions have to be processed in constant time:

- Conditions are O(1) boolean predicates.
- Actions are : (i) Block or transmit the message, (ii) Log information, (iii) Update a local variable,

Thus processing one command only depends on the number of rules:

- For all predicates P, worst case processing time T of a message is $T = \sum \tau_i n_i$
- With τ_i the processing time of predicate P_i
- And n_i number of occurrences of predicate P_i

In practice, as only relevant rules are tested for a message.
Worst-case happens for an accepted message.

Open Secure Channel Sub-Protocol

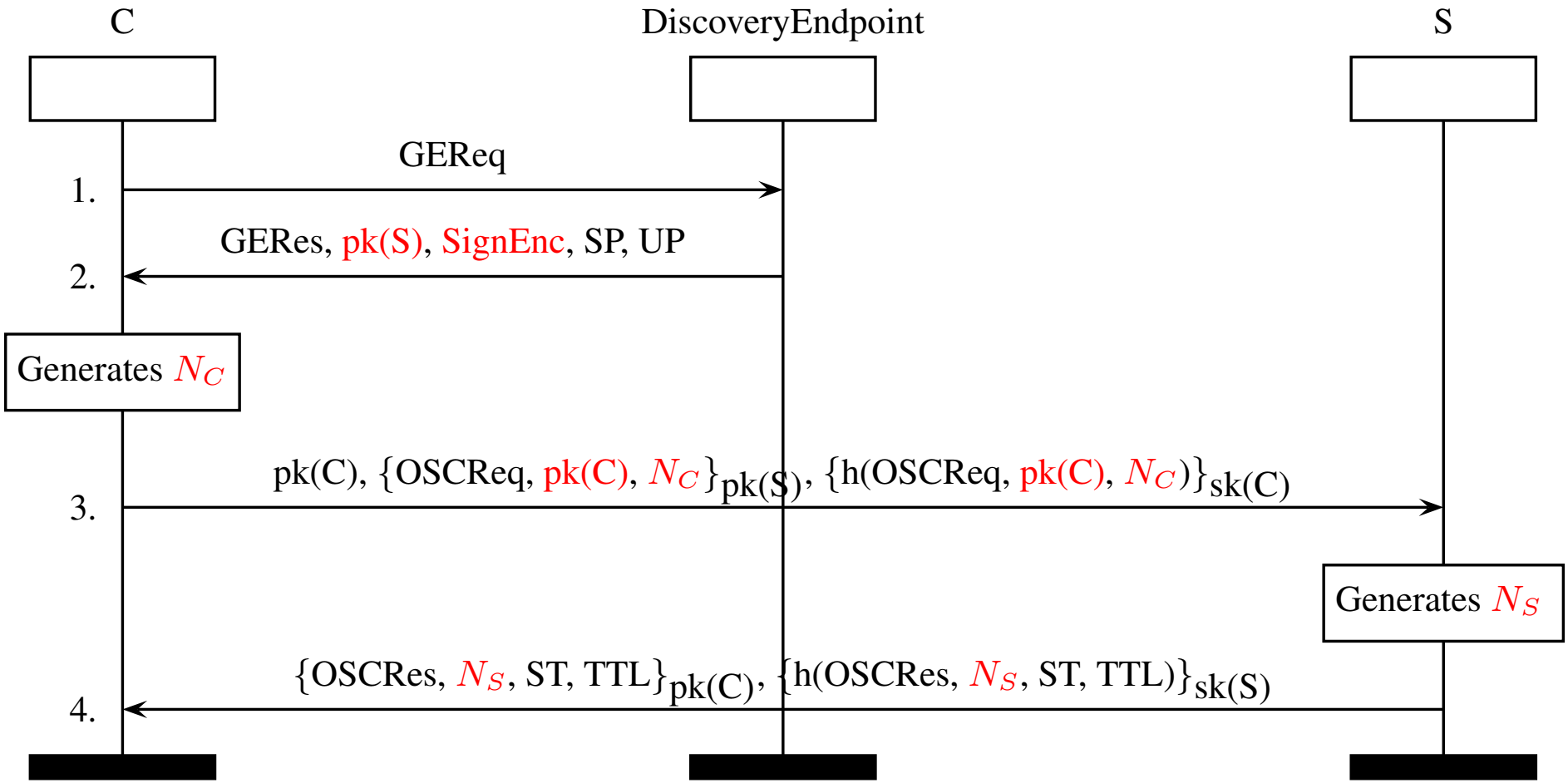


Figure : OPC-UA OpenSecureChannel

Nonce: random value for freshness or challenges/responses.

Modeling Hypotheses

- Normally, several responses to a GetEnpointRequest.
  - We suppose that the client receives and accepts a single one.
  - We tried all possible combinations.
- Client's and server's certificates are modeled by their public keys.
  - Common practice since other fields are out of the scope of tools.
- The intruder can be legitimate clients or servers (e.g.: corrupted devices, malicious operators, etc).
  - Increasing the power of the intruder.
- Objectives:
  - Secrecy of the generated keys (K_{CS}, K_{SC}) from N_C and N_S.
  - Authentication on exchanged nonces N_C and N_S.

Attack on Authentication on N_C in SignAndEncrypt

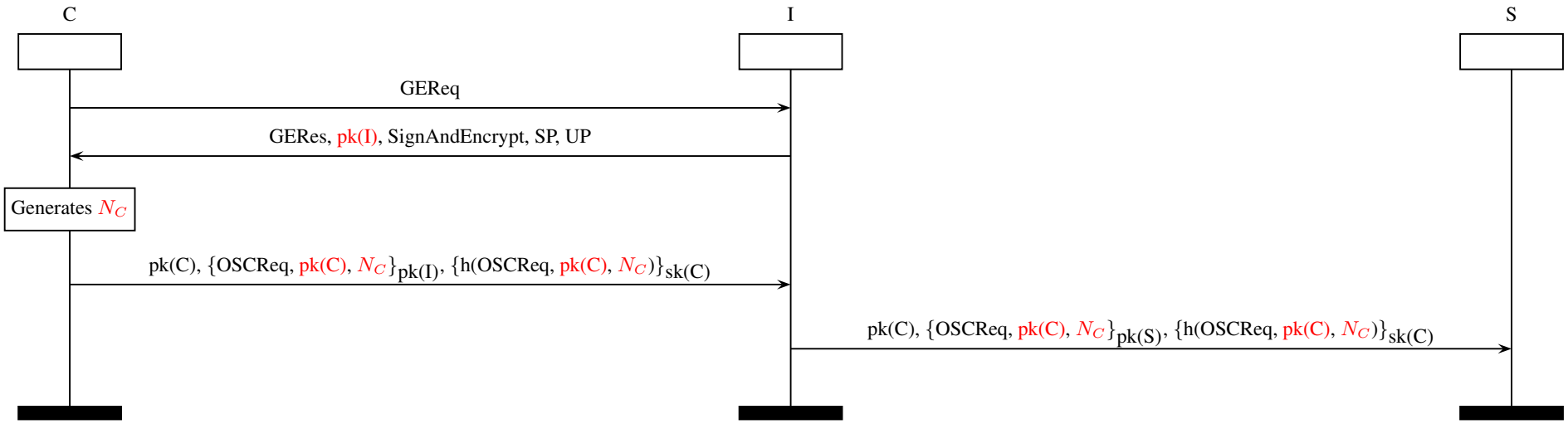


Figure : Attack on OPC-UA OpenSecureChannel

A message can be replayed because receiver is not mentioned in signature.

Create Session Sub-Protocol

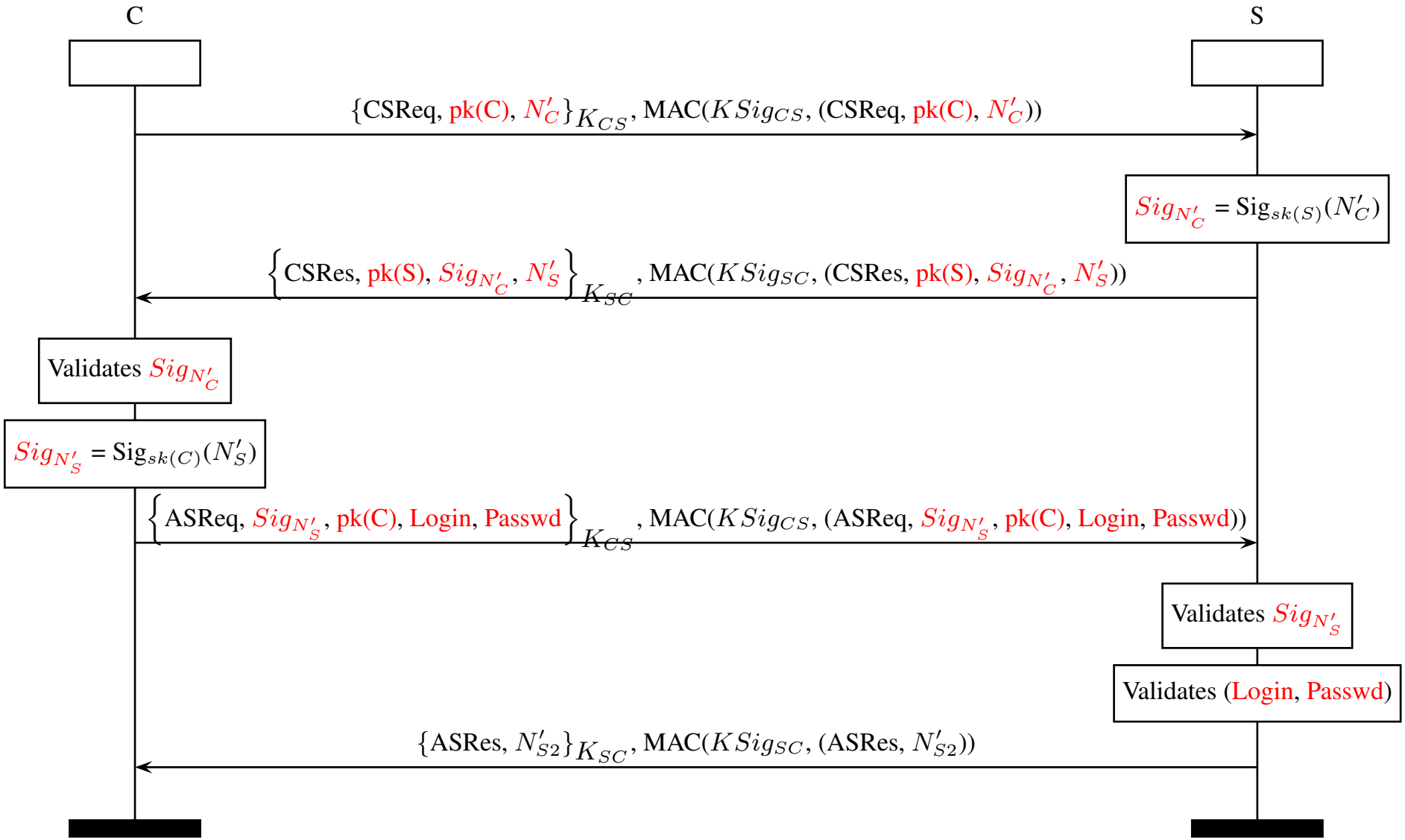


Figure : OPC-UA CreateSession

Non-Injective Message Authenticity (NIMA)

Property

« All messages received have been sent. »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.

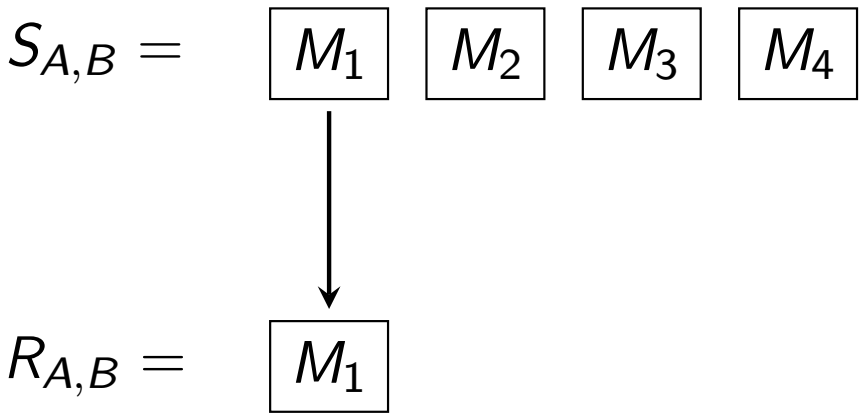
$$S_{A,B} = \boxed{M_1} \boxed{M_2} \boxed{M_3} \boxed{M_4}$$

$$R_{A,B} =$$

Non-Injective Message Authenticity (NIMA)

Property

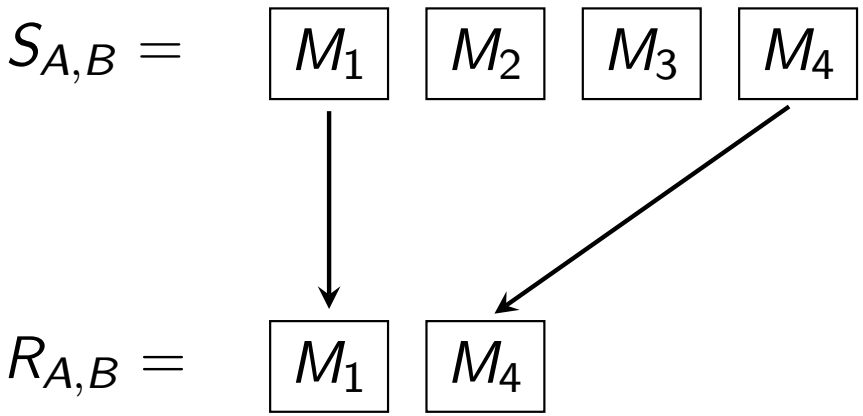
« All messages received have been sent. »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.



Non-Injective Message Authenticity (NIMA)

Property

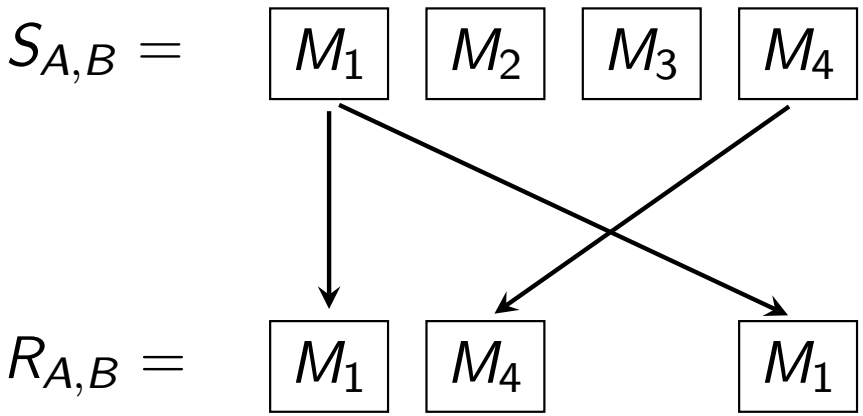
« All messages received have been sent. »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.



Non-Injective Message Authenticity (NIMA)

Property

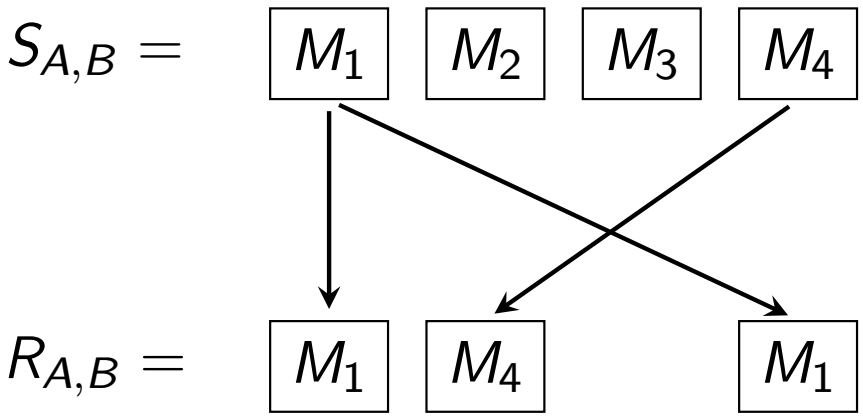
« All messages received have been sent. »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.



Non-Injective Message Authenticity (NIMA)

Property

« All messages received have been sent. »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.

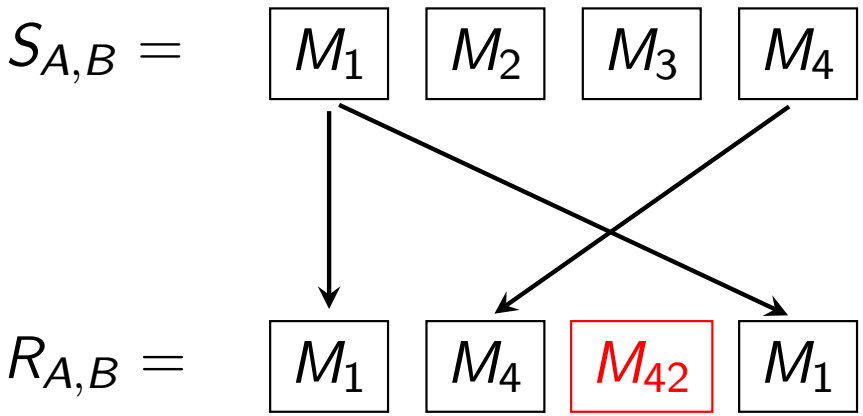


✓ NIMA verified

Non-Injective Message Authenticity (NIMA)

Property

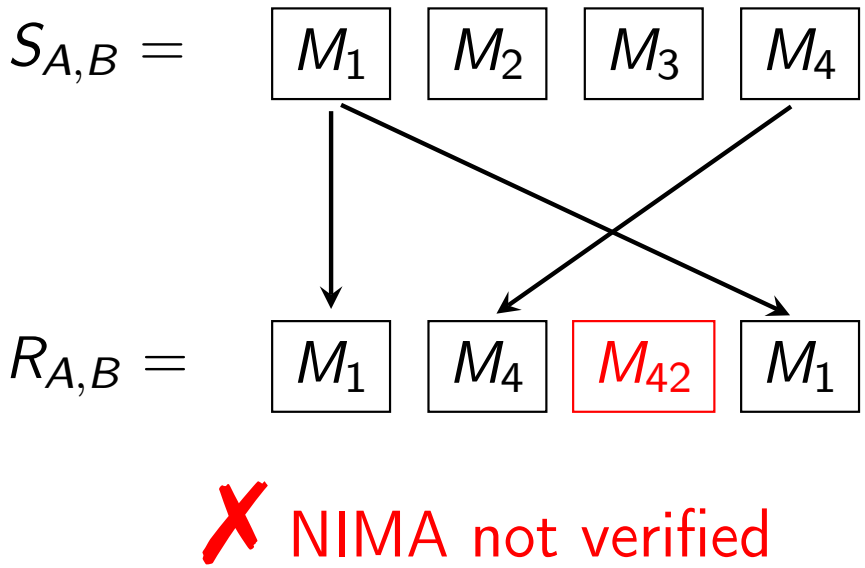
« All messages received have been sent. »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.



Non-Injective Message Authenticity (NIMA)

Property

« **All messages received have been sent.** »
A protocol ensures Non-Injective Message Authenticity (NIMA) between sender A and receiver B if $\textbf{set}(R_{A,B}) \subseteq \textbf{set}(S_{A,B})$.



✗ NIMA not verified

Injective Message Authenticity (IMA)

Property

« **All messages received n times have been sent n times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{\textit{multiset}}(R_{A,B}) \subseteq \textbf{\textit{multiset}}(S_{A,B})$.

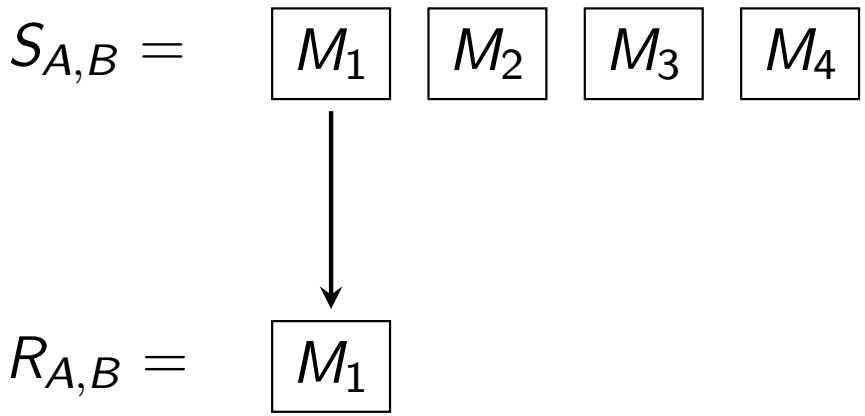
$$S_{A,B} = \boxed{M_1} \boxed{M_2} \boxed{M_3} \boxed{M_4}$$

$$R_{A,B} =$$

Injective Message Authenticity (IMA)

Property

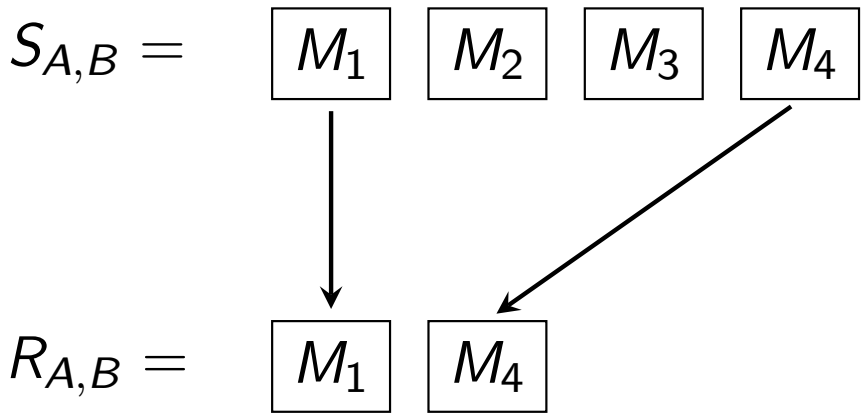
« **All messages received n times have been sent n times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{\textit{multiset}}(R_{A,B}) \subseteq \textbf{\textit{multiset}}(S_{A,B})$.



Injective Message Authenticity (IMA)

Property

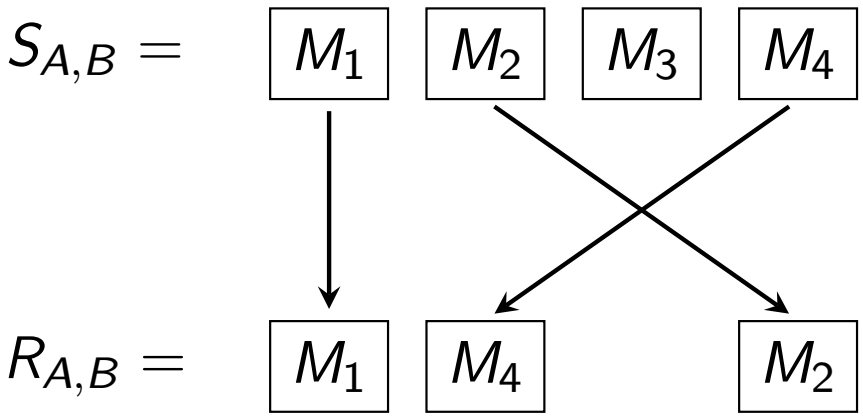
« **All messages received n times have been sent n times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{\textit{multiset}}(R_{A,B}) \subseteq \textbf{\textit{multiset}}(S_{A,B})$.



Injective Message Authenticity (IMA)

Property

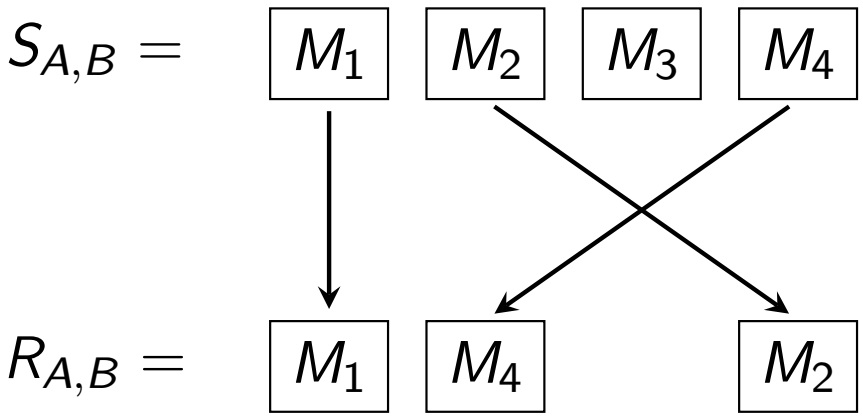
« **All messages received n times have been sent n times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{multiset}(R_{A,B}) \subseteq \textbf{multiset}(S_{A,B})$.



Injective Message Authenticity (IMA)

Property

« **All messages received n times have been sent n times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{\textit{multiset}}(R_{A,B}) \subseteq \textbf{\textit{multiset}}(S_{A,B})$.

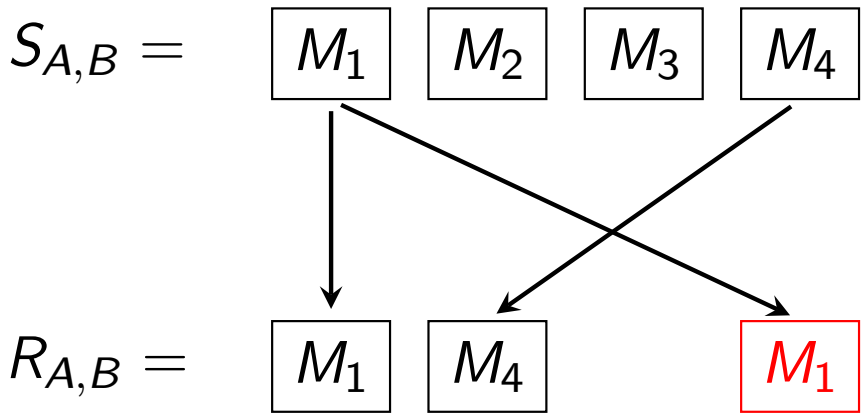


✓ IMA verified

Injective Message Authenticity (IMA)

Property

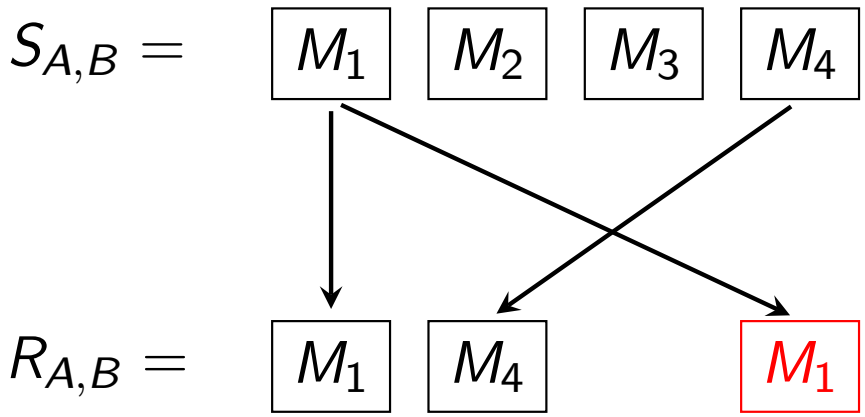
« **All messages received** n **times have been sent** n **times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{\textit{multiset}}(R_{A,B}) \subseteq \textbf{\textit{multiset}}(S_{A,B})$.



Injective Message Authenticity (IMA)

Property

« **All messages received n times have been sent n times.** »
A protocol ensures Injective Message Authenticity (IMA) between sender A and receiver B if $\textbf{\textit{multiset}}(R_{A,B}) \subseteq \textbf{\textit{multiset}}(S_{A,B})$.



✗ IMA not verified

Flow Authenticity (FA)

Property

« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.

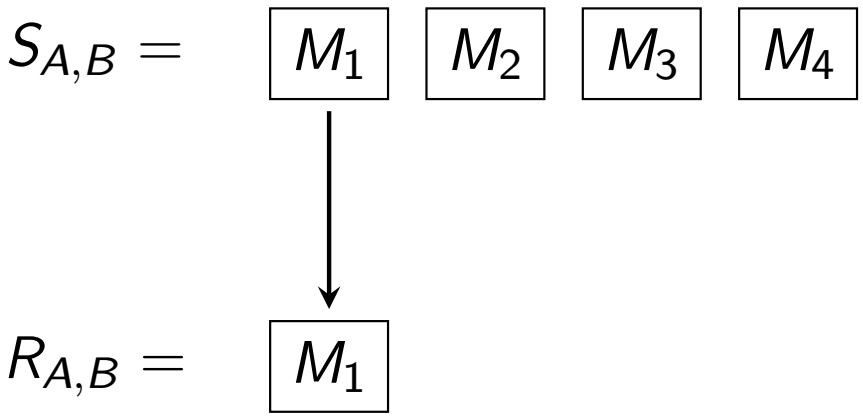
$$S_{A,B} = \boxed{M_1}\ \boxed{M_2}\ \boxed{M_3}\ \boxed{M_4}$$

$$R_{A,B} =$$

Flow Authenticity (FA)

Property

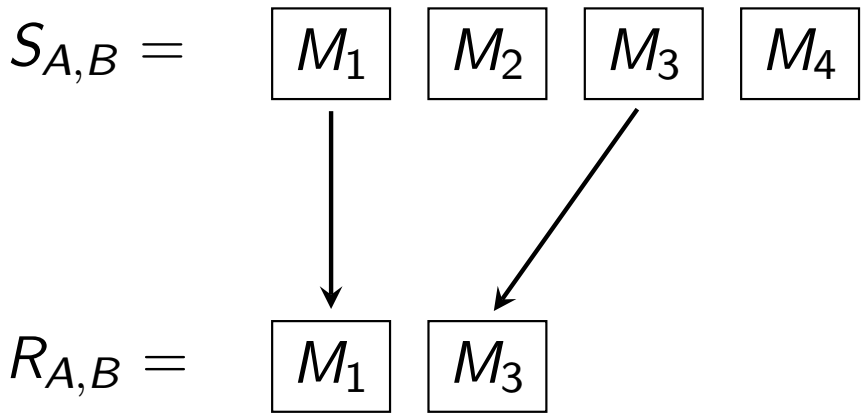
« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.



Flow Authenticity (FA)

Property

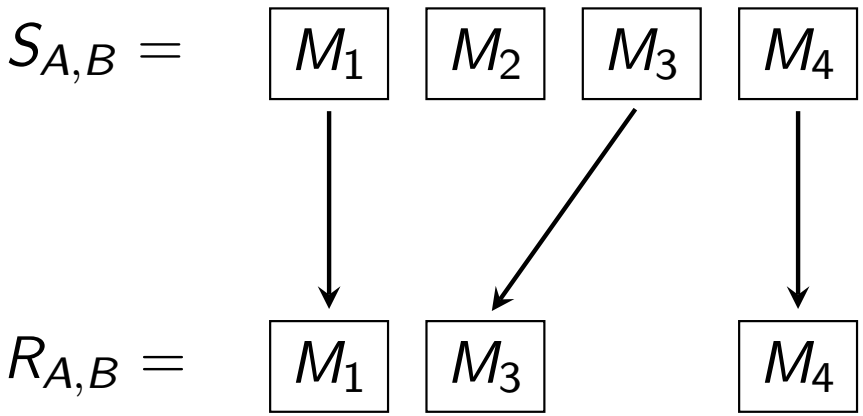
« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.



Flow Authenticity (FA)

Property

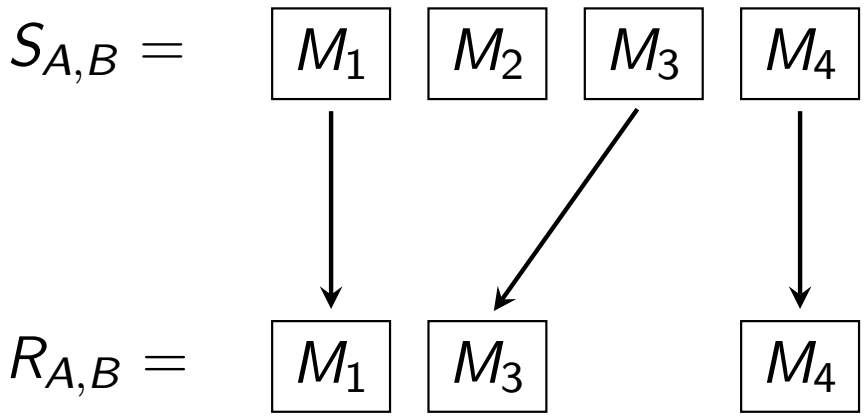
« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.



Flow Authenticity (FA)

Property

« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.

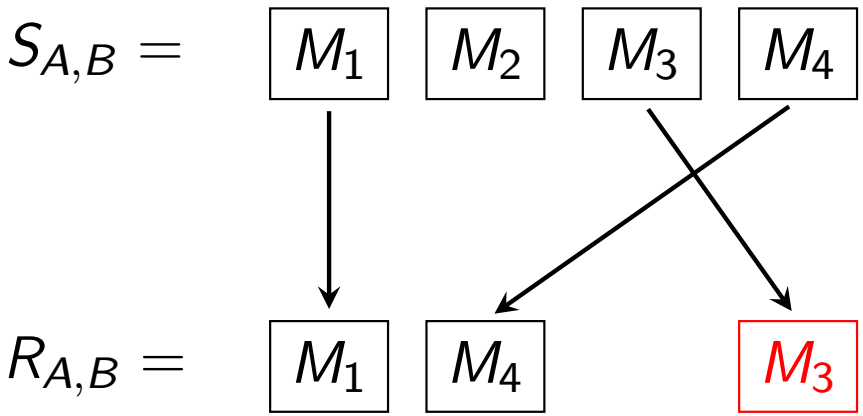


✓ FA verified

Flow Authenticity (FA)

Property

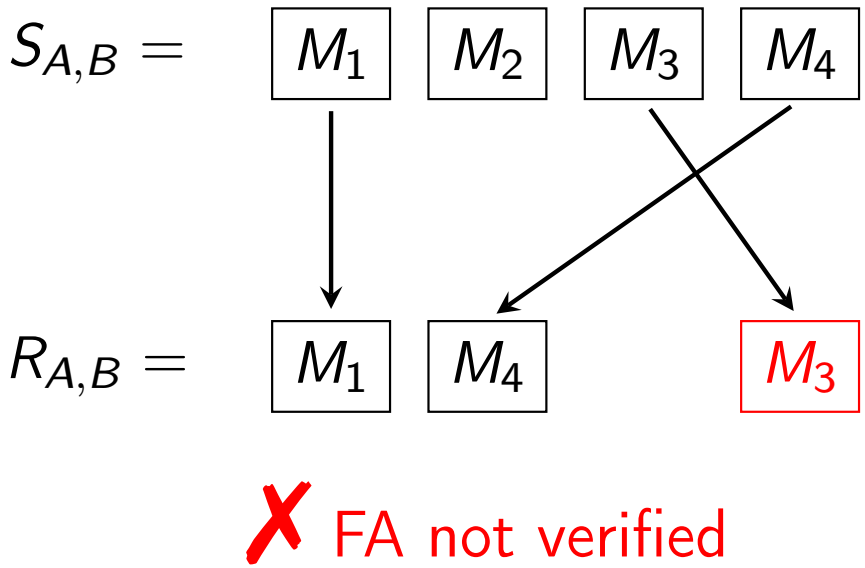
« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.



Flow Authenticity (FA)

Property

« **All messages are received in the order they have been sent.** »
A protocol ensures Flow Authenticity (FA) between sender A and receiver B if $R_{A,B}$ **is a sub-chain of** $S_{A,B}$.



Non-Injective Message Authenticity (NIMA)

Property

« All messages received have been sent. »

$$\forall i : time, A, B : agent, m : msg.$$
$$Received(A, B, m)@i \Rightarrow ($$
$$\exists j : time.Sent(A, B, m)@j \wedge j \lessdot i$$
$$)$$

Injective Message Authenticity (IMA)

Property

« All messages received n times have been sent n times. »

$$
\begin{aligned}
&\forall i : \mathit{time}, A, B : \mathit{agent}, m : \mathit{msg}.\\
&\quad \mathit{Received}(A, B, m)@i \Rightarrow (\\
&\qquad \exists j.\mathit{Sent}(A, B, m)@j \wedge j \lessdot i \wedge \neg(\\
&\qquad\quad \exists i2 : \mathit{time}, A2, B2 : \mathit{agent}.\\
&\qquad\quad \mathit{Received}(A2, B2, m)@i2 \wedge \neg(i2 \doteq i)\\
&\qquad)\\
&\quad)
\end{aligned}
$$

Flow Authenticity (FA)

Property

« All messages are received in the same order they have been sent. »

$$\forall i, j : time, A, B : agent, m, m_2 : msg.($$
$$\quad Received(A, B, m)@i \wedge Received(A, B, m_2)@j \wedge i \lessdot j$$
$$) \Rightarrow (\exists k, l : time.$$
$$\quad Sent(A, B, m)@k \wedge Sent(A, B, m_2)@l \wedge k \lessdot l$$
$$)$$

Resilient Channels

- Dolev-Yao intruder can block message, thus delivery is always false!
- Enforce intruder that all messages are eventually delivered.
- Security properties do not hold vacuously (still allows duplicating, reordering, delaying, forging).

$$\forall i : time, m : msg.Ch\_Sent(m)@i$$
$$\Rightarrow (\exists j.Ch\_Received(m)@j \wedge i \lessdot j)$$

Top-Down Example

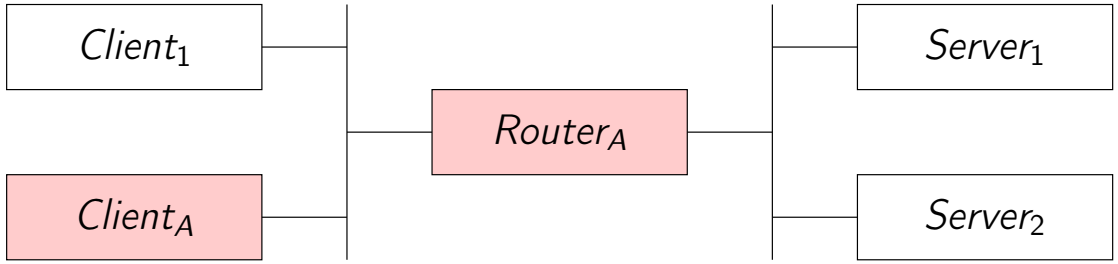


Figure : Infrastructure example

Possible security objectives:

- $IdTh$ = Identity theft,
- $AuthBP$ = Authentication by-pass,

$\mathcal{R}_{Obj}$	$IdTh$	$AuthBP$
$Client_A$	✗	✓
$Router_A$	✓	✗

Table : Objectives for each attacker

Bottom-Up Example

Possible realization of objectives:

- $Real(IdTh) = \{\{Spy\}\}$
- $Real(AuthBP) = \{\{Usurp\}, \{Replay\}\}$

$Atk.vectors$	Spy	$Usurp$	$Replay$
FTP_{Auth}	✓	✗	✓
$\text{OPC-UA}_{SignEnc}$	✗	✗	✗

Table : Atk. vectors for each protocol

Results:

- $\mathcal{S}_{Client_A,\text{FTP}_{Auth}} = \{(AuthBP, Replay)\}$
- $\mathcal{S}_{Client_A,\text{OPC-UA}_{SignEnc}} = \emptyset$
- $\mathcal{S}_{Router_A,\text{FTP}_{Auth}} = \{(IdTh, Spy)\}$
- $\mathcal{S}_{Router_A,\text{OPC-UA}_{SignEnc}} = \emptyset$

Clients and Servers

For a transport protocol:

- Encapsulate and decapsulate applicative message into packets.
- Reusable for a model to another.
- BehaviorClient generates applicative messages.
- SecurityLayer performs cryptographic operations.