Control Flow Analysis for process algebras with applications to security

Chiara Bodei[1] and many others

Dip. di Informatica, Università di Pisa, Italy

SKIP TO SLIDE 53

Outline

1. CFA applied to Pi Calculus
2. Background on security protocols
3. CFA and security protocols
4. Possible further directions of investigation
5. CFA and Internet of Things
6. Conclusions

Static Analysis: why?

- There are many questions we can ask about a given program.
- all the interesting questions about the behaviour of a program are undecidable, **but**
- we want to solve practical questions

$\Rightarrow$ approximate answers, still precise enough to fuel our applications.
Approximations are conservative: all the errors lean to the same side

Efficiency Concerns

- Transition systems: usually huge $\Rightarrow$ their exploration can be computationally hard.
- Need of obtaining information about the dynamic behaviour, without spending so much.
- We can look at the system description: static analysis techniques

Static Analysis Techniques

Static techniques
- Abstract Interpretation
- Control Flow Analysis (CFA)
- Type Systems

- Non trivial information about the dynamic behaviour, by simply inspecting the description of the system.
- predict safe and computable approximations of dynamic behaviuor
- analyse properties that hold in every execution
- give a repertoire of automatic and decidable methods and tools

In checking properties

STATICALLY	*DYNAMICALLY*
(analyse the TEXT)	(analyse the BEHAVIOUR)
approximate	precise
terminates	may not terminate
low complexity	high complexity
cheap tools	expensive tools

SOUNDNESS

P has the static property implies P has the dynamic property

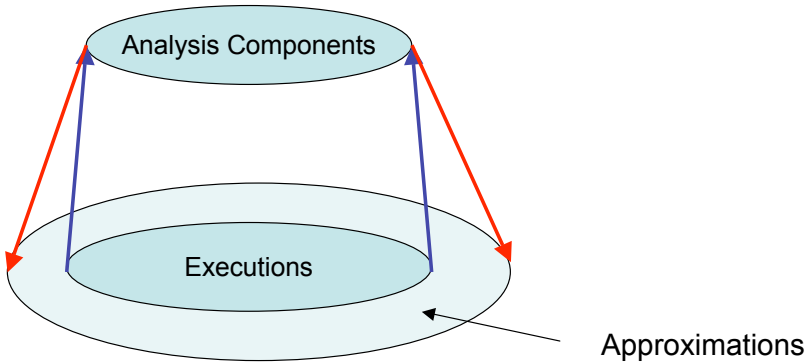
Control Flow Analysis (CFA)

- It predicts approximations (estimates) to the set of values that the objects of a program may assume at run-time.
- It treats
  - semantic correctness
  - existence of least estimates
  - efficient construction of least estimates

1. The analysis collects data and afterwards,
2. it checks them in order to prove properties

Control Flow Analysis (CFA)

- CFA represents an abstraction of the actual executions
- Concretisation cannot be precise.



Static	Dynamic
Event E is included	E can happen
Event E is not included	E never happens

CFA pattern

- Choose those values of interest for the language
  - Define the shape of estimates
  - Define a number of clauses
- Prove that all estimates are semantically correct
- Prove that least estimate exist
- Derive a constructive procedure that builds estimates
- Select a specific dynamic security property and define a static check on estimates

CFA vs Type Systems

- Type Systems are prescriptive, i.e. they infer types and impose the well-formedness conditions at the same time.
- Control Flow Analysis is mainly descriptive, i.e. it merely infers the information and then leaves it to a separate step to actually impose demands on when programs are well-formed.
- For each property, it is often the case that:
  - a new *ad hoc* type system is necessary, while
  - only a new test on the *same* CFA analysis is needed.

Pi calculus: Remind

- **Communication** primitives: simple and powerful.
- **Scoping Rules**: explicitly control the access to channels and to data.

Processes

$$P ::= \mathbf{0} \mid \tau.P \mid x(y).P \mid \overline{x}y \mid (\nu x)P \mid [x{=}y]P \mid P \mid P \mid !P \mid P + P$$

Communication semantic rule

$$\frac{P \xrightarrow{x(a)} P', Q \xrightarrow{\overline{x}a} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

Abstract Behaviour

How is abstract behaviour described? As a pair of abstract domains given as functions

$$(\rho, \kappa)$$

ρ : name $\mapsto$ {set of names} it can be bound to, and $\rho(n) = \{n\}$ for every free name n.
κ : binder/name $\mapsto$ {set of names} that can be sent over it.

Abstract Behaviour: example

$$(P \mid Q \mid R) = (\underbrace{\overline{a}d.P' + \overline{a}b.P''}_{P} \mid R \mid \underbrace{a(w).\overline{c}w.Q'}_{Q})$$

This process can evolve as:

- $P' \mid R \mid \overline{c}d.Q'$
- $P'' \mid R \mid \overline{c}b.Q'$

In the first case w becomes d, while in the second one becomes b.
The analysis must predict **both** possibilities:

$$(\rho(w) \supseteq \{b, d\}$$

Abstract Behaviour: example

$$P \mid Q \mid R = (\underbrace{\overline{a}d.P' + \overline{a}b.P''}_{P} \mid R \mid \underbrace{a(w).\overline{c}w.Q'}_{Q})$$

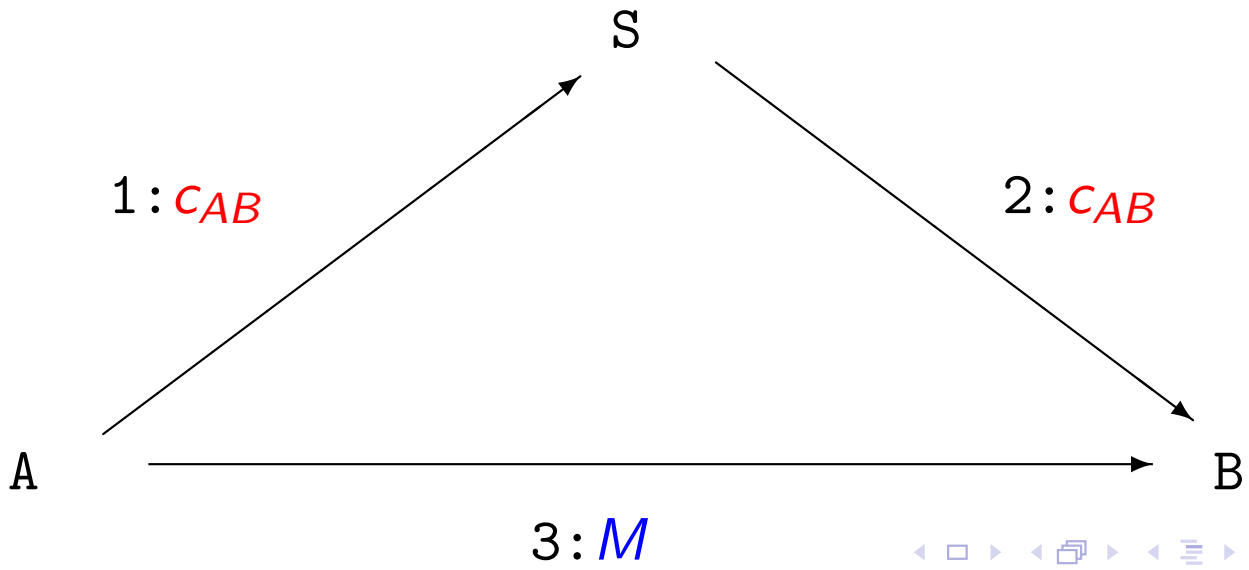
	ρ
a	$\{a\}$
b	$\{b\}$
c	$\{c\}$
d	$\{d\}$
w	$\{b, d\}$

	κ
a	$\{b, d\}$
b	$\emptyset$
c	$\{b, d\}$
d	$\emptyset$

CFA Example: Wide Mouthed Frog key exchange protocol

(See Section Background on security protocols)

$$
\begin{aligned}
P &= (\nu c_{AS})(\nu c_{BS})(\,(A|B)\,|S) \\
A &= (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S &= c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B &= c_{BS}(w).w(y)
\end{aligned}
$$



CFA Example: Wide Mouthed Frog (2)

$$
\begin{aligned}
P &= (\nu c_{AS})(\nu c_{BS})(\ (A|B)\ |S) \\
A &= (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S &= c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B &= c_{BS}(w).w(y)
\end{aligned}
$$

$$
\begin{array}{llll}
\rho(x) \supseteq \{c_{AB}\} & & \kappa(c_{AS}) \supseteq \{c_{AB}\} \\
\rho(w) \supseteq \{c_{AB}\} & & \kappa(c_{BS}) \supseteq \{c_{AB}\} \\
\rho(y) \supseteq \{M\} & & \kappa(c_{AB}) \supseteq \{M\}
\end{array}
$$

Estimates Validation

When is (ρ, κ) a *valid* estimate? Formally:

$$(\rho, \kappa) \models P$$

When it satisfies a set of logical clauses (one for each syntactic case). Zoom on the principal ones:

$$(\rho, \kappa) \models \overline{x}\langle y \rangle.P \textbf{ iff } (\rho, \kappa) \models P \wedge \boxed{\forall a \in \rho(x) : \rho(y) \subseteq \kappa(a)}$$

The set of names that can be communicated along x includes the names to which y can evaluate ro (i.e. those in $\rho(y)$).

$$(\rho, \kappa) \models x(y).P \text{ iff} (\rho, \kappa) \models P \wedge \boxed{\forall a \in \rho(x), \kappa(a) \subseteq \rho(y)}$$

The set of channels that can be bound to y includes those that can be communicated along x.

Estimates Validation

$$(\rho, \kappa) \models \mathbf{0} \text{ iff } \mathit{true}$$

$$(\rho, \kappa) \models (\nu x)P \text{ iff } (\rho, \kappa) \models P$$

$$(\rho, \kappa) \models P \mid Q \text{ iff } (\rho, \kappa) \models P \wedge (\rho, \kappa) \models Q$$

$$(\rho, \kappa) \models P + Q \text{ iff } (\rho, \kappa) \models P \wedge (\rho, \kappa) \models Q$$

$$(\rho, \kappa) \models !P \text{ iff } (\rho, \kappa) \models P$$

Note that the analysis does not consider restriction and replication: this is one of the sources of approximation.

Estimates Validation

$$(\rho, \kappa) \models [x = y]P \text{ iff } \rho(x) \cap \rho(y) \neq \emptyset \Rightarrow (\rho, \kappa) \models P$$

Note that the check proceeds only if the match can be successful, otherwise there is no point in analysing a sub-process that is not **reachable**.

The Nature of Imprecision

This Control Flow Analysis is Context-Insensitive and is called 0-CFA.

$$P_1 = (a(y)|\ \overline{a}b) \qquad P_2 = (a(y) + \overline{a}b) \qquad P_3 = (a(y).\overline{a}b)$$

- Note that the variable y in P_1 can be bound to b, while in P_2 and in P_3 it cannot.
- Instead, in the CFA estimate, we have that $\rho(y) \supseteq \{b\}$ in all the three cases.

Back to the example

$$
\begin{array}{lcl}
P & = & (\nu c_{AS})(\nu c_{BS})(\ (A|B)\ |S) \\
A & = & (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S & = & c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B & = & c_{BS}(w).w(y)
\end{array}
$$

$$
\begin{array}{llll}
\rho(x) \supseteq \{c_{AB}\} & & \kappa(c_{AS}) \supseteq \{c_{AB}\} \\
\rho(w) \supseteq \{c_{AB}\} & & \kappa(c_{BS}) \supseteq \{c_{AB}\} \\
\rho(y) \supseteq \{M\} & & \kappa(c_{AB}) \supseteq \{M\}
\end{array}
$$

- $(\rho,\kappa) \models P \Leftrightarrow (\rho,\kappa) \models A \wedge (\rho,\kappa) \models S \wedge (\rho,\kappa) \models B$
- $(\rho,\kappa) \models A \Leftrightarrow (\rho,\kappa) \models \overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle \Leftrightarrow$
  $\{c_{AB}\} \subseteq \kappa(c_{AS}) \wedge (\rho,\kappa) \models \overline{c_{AB}}\langle M\rangle \Leftrightarrow$
  $\{c_{AB}\} \subseteq \kappa(c_{AS}) \wedge \{M\} \subseteq \kappa(c_{AB})$
- $(\rho,\kappa) \models S \Leftrightarrow (\rho,\kappa) \models c_{AS}(x).\overline{c_{BS}}\langle x\rangle \Leftrightarrow \{c_{AB}\} \subseteq \rho(x) \wedge (\rho,\kappa) \models \overline{c_{BS}}\langle x\rangle \Leftrightarrow$
  $\{c_{AB}\} \subseteq \rho(x) \wedge \{c_{AB}\} \subseteq \kappa(c_{BS})$
- $(\rho,\kappa) \models B \Leftrightarrow (\rho,\kappa) \models c_{BS}(w).w(y) \Leftrightarrow \{c_{AB}\} \subseteq \rho(w) \wedge (\rho,\kappa) \models w(y) \Leftrightarrow$
  $\{c_{AB}\} \subseteq \rho(w) \wedge \{M\} \subseteq \rho(y)$

CFA Properties

SOUNDNESS

The analysis is semantically correct. If $(\rho, \kappa) \models P$ and $P \rightarrow Q$ then $(\rho, \kappa) \models Q$

EXISTENCE

- The set of proposed solutions can be partially ordered ($(\rho, \kappa) \subseteq (\rho', \kappa')$ iff ordered pointwise).
- The set $\{(\rho, \kappa) \mid (\rho, \kappa) \models P\}$ is a Moore family*
- Therefore there always exists a least solution (ρ, κ).

(*) A set $\mathcal{I}$ of proposed estimates is a *Moore family* if and only if it contains $\sqcap \mathcal{J}$ for all $\mathcal{J} \subseteq \mathcal{I}$.

CFA Properties (cont.)

CONSTRUCTION

There is a constructive procedure for obtaining the least solution of low polynomial complexity.

CFA pattern (2)

- Choose those values of interest for the language and define estimates and clauses.
- Prove that all estimates are semantically correct.
- Prove that least estimate exist.
- Derive a *constructive* procedure that builds estimates.

CFA pattern (2)

- Choose those values of interest for the language and define estimates and clauses.
- Prove that all estimates are semantically correct.
- Prove that least estimate exist.
- Derive a *constructive* procedure that builds estimates.
- Select a specific dynamic security property and define a static check on estimates.

This may require to refine estimates.

Now comes Security: a simple **Secrecy** property

Dynamic notion: Carefulness

No secret datum flows on a public channel.

Static notion: Confinement

1. partition names in $\begin{cases} \mathcal{S} & \text{Secret names} \\ \mathcal{P} & \text{Public names} \end{cases}$
2. compute $(\rho, \kappa) \models P$
3. check that $\kappa(a \in \mathcal{P}) \subseteq \mathcal{P}$

Theorem If a system S is confined then S is careful.

Example

$$
\begin{aligned}
P &= (\nu c_{AS})(\nu c_{BS})(\,(A|B)\,|S) \\
A &= (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S &= c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B &= c_{BS}(w).w(y)
\end{aligned}
$$

$$
\begin{array}{llll}
\rho(x) \supseteq \{c_{AB}\} & & \kappa(c_{AS}) \supseteq \{c_{AB}\} & \\
\rho(w) \supseteq \{c_{AB}\} & & \kappa(c_{BS}) \supseteq \{c_{AB}\} & \\
\rho(y) \supseteq \{M\} & & \kappa(c_{AB}) \supseteq \{M\} & \leftarrow
\end{array}
$$

If $\mathcal{S} = \{M, c_{AS}, c_{BS}\}$ and $\mathcal{P} = \{c_{AB}\}$, then P has leaks:

$\mathcal{S} \ni M \in \kappa(c_{AB} \in \mathcal{P}) \not\subseteq \mathcal{P}$

The process would have no leaks if c_{AB} and M, were secret or public, respectively.

Other example

$$P \mid Q \mid R = (\underbrace{\overline{a}d.P' + \overline{a}b.P''}_{P} \mid R \mid \underbrace{a(w).\overline{c}w.Q'}_{Q})$$

Suppose that $\mathcal{S} = \{a, d\}$ and that $\mathcal{P} = \{b, c\}$.

	ρ
a	$\{a\}$
b	$\{b\}$
c	$\{c\}$
d	$\{d\}$
w	$\{b, d\}$

	κ
a	$\{b, d\}$
b	$\emptyset$
c	$\{b, d\}$
d	$\emptyset$

The system does not satisfy the property: $\boxed{\kappa(c) = \{b, d\} \not\subseteq \mathcal{P}}$

Another Property: No Read Up/No Write Down (NRU/NWD)

Processes are given levels of security clearance

NRU/NWD: the sender has a clearance level lower than the level of the receiver.

- Syntax: $S ::= \langle P \rangle^l \mid (\nu x)S \mid S|S \mid !S$,with l level label;

Another Property: No Read Up/No Write Down (NRU/NWD)

Processes are given levels of security clearance

NRU/NWD: the sender has a clearance level lower than the level of the receiver.

- Syntax: $S ::= \langle P \rangle^{l} \mid (\nu x)S \mid S|S \mid !S$,with l level label;
- Semantics: $$\frac{P \xrightarrow{\mu} Q}{\langle P \rangle^{l} \xrightarrow{\mu,l} \langle Q \rangle^{l}}$$

Another Property: No Read Up/No Write Down (NRU/NWD)

Processes are given levels of security clearance
NRU/NWD: the sender has a clearance level lower than the level of the receiver.

- Syntax: $S ::= \langle P \rangle^l \mid (\nu x)S \mid S|S \mid !S$,with l level label;
- Semantics: $$\frac{P \xrightarrow{\mu} Q}{\langle P \rangle^l \xrightarrow{\mu,l} \langle Q \rangle^l}$$
- Analysis: $(\rho, \kappa, \sigma) \models_l P$, with $\sigma = \langle \sigma_{in}, \sigma_{out} \rangle$, where
  - $\sigma_{in}(l)$: set of the channels that can be received by an input within a sub-process with level l.
  - $\sigma_{out}(l)$: set of the channels that can be sent by an output within a sub-process with level l.

NRU/NWD: CFA rules

- $(\hat{\Sigma}, \kappa) \models l\overline{x}y.P$ iff $(\hat{\Sigma}, \kappa) \models lP \wedge$
$$\forall a \in \rho(x) : \begin{pmatrix} \rho(y) \subseteq \kappa(a) \wedge \\ \rho(y) \subseteq \sigma_{out}(l)(a) \end{pmatrix}$$
- $(\hat{\Sigma}, \kappa) \models lx(y).P$ iff $(\hat{\Sigma}, \kappa) \models lP \wedge$
$$\forall a \in \rho(x) : \begin{pmatrix} \kappa(a) \subseteq \rho(y) \wedge \\ \kappa(a) \subseteq \sigma_{in}(l)(x) \end{pmatrix}$$
- $(\hat{\Sigma}, \kappa) \models \langle P \rangle^l$ iff $(\hat{\Sigma}, \kappa) \models lP$

NRU/NWD

Dynamic Property: **no read-up/no write-down**

A high level process cannot write any value to a process at low level; symmetrically a process at low level cannot read data from one of a high level.

Static Property: **discreet**

Each channel cannot be used for sending an object from a process with high level l to a process with low level l'.

$$\forall l', l \text{ with } l' \text{ below } l : \forall a : \sigma_{out}(l)(a) \cap \sigma_{in}(l')(a) = \emptyset.$$

Theorem If a system S is discreet then S is No Read Up/No Write Down.

Bibliography

- Chiara Bodei, Pierpaolo Degano, Flemming Nielson and Hanne Riis Nielson. *Static Analysis for the Pi-Calculus with Applications to Security*. Information and Computation, 168: 68-92, 2001.
- Hanne Riis Nielson, Flemming Nielson, Henrik Pilegaard. *Flow Logic for Process Calculi*. ACM Comput. Surv. 44(1): 3 (2012).

Security protocols: why?

- There are many cryptographic primitives
- How can we use them to secure applications?
- It is a difficult task, even if cryptography is "perfect"

Security protocols

- A security protocol is a distributed algorithm. It consists of a set of rules (conventions) that determine the exchange of messages between two or more principals.
- Since principals communicate over channels an untrusted network, security protocols ensure that communication is not abused

Security protocols (cont.)

> *Security protocols are three-line programs that people still manage to get wrong.*
>
> *Roger M. Needham*

We focus in particular on authentication protocols

Problem: how to establish a virtual trusted channel:

- negotiate parameters of channel
- ensure that channel is still trusted

Security protocols (cont.)

Main characters are

- generic principals (aka agents, parties...): A (Alice), B (Bob)
- a trusted server: S (Sam)
- the **Dolev-Yao** attacker: C (Charlie) or M (Mallory - Malicious) or E (Eve - Evil) (it can eavesdrop, replay, encrypt, decrypt, generate and inject messages.)

A protocol involves a sequence of message exchanges of the form:

$$A \rightarrow B : Msg$$

meaning that a principal A sends the message Msg to principal B
$C(A)$ stands for C acting as A

Security protocols (cont.)

- Is cryptography enough?
- Attacks can be successful even without attacking the crypto-keys.

Wolf-in-the-middle attack: the wolf does not have the key

1. Little Red Riding Hood → Grandma

The wolf does not need the key: grandma opens the door

1′ Wolf(Little Red Riding Hood) → Grandma

2. Grandma → Little Red Riding Hood

2′. Wolf(Grandma) → Little Red Riding Hood

Little Red Riding Hood believes grandma opens the door

Bank transfer protocol

Example (Naive version)

$Alice \rightarrow Bob@Bank$: Transfer 100 euros to account X
$Bob@Bank \rightarrow Alice$: Tranfer just carried out

- How does Bob know that he is really speaking with Alice?
- How does Bob know Alice just said it?
- Bad guys like Charlie can be around

Bank transfer protocol (cont.)

Example (Easy attack)

$Alice \rightarrow Charlie(Bob@Bank)$: Transfer 100 euros to account X
$Charlie(Bob@Bank) \rightarrow Bob@Bank$: Transfer 200 euros to account X
$Bob@Bank \rightarrow Alice$: Tranfer just carried out

- Charlie can intercept and alter the message of Alice

$C(B)$ stands for C pretending to be B.

Bank transfer protocol (cont.)

Example

Cryptographic protection

$$Alice \rightarrow Charlie(Bob@Bank) : \{\text{Transfer 100 euros to account X}\}_K$$

...

- The attacker can intercept the message of Alice, but cannot alter it

Bank transfer protocol (cont.)

Example (Cryptography may not be not enough: reflection attack)

$$Alice \rightarrow Charlie(Bob@Bank) : \{\text{Transfer 100 euros to account X}\}_K$$
$$Charlie(Bob@Bank) \rightarrow Alice : \{\text{Transfer 100 euros to account X}\}_K$$

- The attacker can intercept the message of Alice, and just use it again
- Alice does not realize the received message is the one she generated

Bank transfer protocol (cont.)

Example (Possible fix)

$Alice \rightarrow Bob@Bank : \{\text{I am Alice, Transfer 100 euros to account X}\}_K$
$Bob@Bank \rightarrow Alice : \text{Tranfer just carried out}$

Bank transfer protocol (cont.)

Example (Replay attack)

$Alice \rightarrow Bob@Bank : \{\text{I am Alice, Transfer 100 euros to account X}\}_K$
$Charlie \rightarrow Bob@Bank : \{\text{I am Alice, Transfer 100 euros to account X}\}_K$

- The attacker can intercept the message of Alice, and just use it again
- Bob does not realize the received message is an old one: he has no way to verify the freshness of the message

Bank transfer protocol (cont.)

Example (Fixed protocol)

$Bob@Bank \rightarrow Alice : N_B$
$Alice \rightarrow Bob@Bank : \{\text{I am Alice, Transfer 100 euros to account X}, N_B\}_K$
...

- Use a nonce, a randomly generated number used only once.
- Principals do not see each other: their trust is based on the presence of expected and known terms in received messages.
- Bob can verify that the message is an answer to his request

This protocol is secure. It guarantees the secrecy and authenticity of the message

Security protocols (cont.)

- Is cryptography enough?
- Attacks can be successful even without attacking the crypto-keys.
- Cryptography translates a communication security problem into a key management problem; therefore, it can enhance security but it is not a substitute for security

An authentication protocol

- Security protocol: list of messages exchanged by principals include recurring terms, like *rhymes* in poem verses

Example (Needham-Schroeder Public key protocol)

1. $A \to B : \quad \{N_A, A\}_{pk(K_B)}$
2. $B \to A : \quad \{N_A, N_B\}_{pk(K_A)}$
3. $A \to B : \quad \{N_B\}_{pk(K_B)}$

- Security goal: mutual authentication of A and B,
- Principals do not see each other: their trust is based on the presence of expected and known terms in received messages.
- Exchange of messages: information + confirmation

Man-in-the-middle attack

- Attackers can forge messages that are accepted by legitimate parties, breaking the intended rhymes and leading to attacks.

Example (Lowe Attack)

$$
\begin{aligned}
&1.\ A \to M : && \{N_A, A\}_{pk(K_M)} \\
&1'.M_A \to B : && \{N_A, A\}_{pk(K_B)} \\
&2'.B \to M_A : && \{N_A, N_B\}_{pk(K_A)} \\
&2.\ M \to A : && \{N_A, N_B\}_{pk(K_A)} \\
&3.\ A \to M : && \{N_B\}_{pk(K_M)} \\
&3'\ M_A \to B : && \{N_B\}_{pk(K_B)}
\end{aligned}
$$

where M_X stands for M pretending to be X.

B believes to talk with A, while it is talking with M: B cannot authenticate A.

Fixed protocol

- Confirmation part must be enriched: msg 2) does not say who is the sender

Example

$$\begin{array}{lll} 1. & A \to B : & \{N_A, A\}_{pk(K_B)} \\ 2. & B \to A : & \{N_A, N_B, B\}_{pk(K_A)} \\ 3. & A \to B : & \{N_B\}_{pk(K_B)} \end{array}$$

- the attack is no longer possible

$$\begin{array}{l} \ldots \\ 2'.B \to M_A : \{N_A, N_B, B\}_{pk(K_A)} \\ 2.\ M \to A : \{N_A, N_B, B\}_{pk(K_A)} \end{array}$$

because A is waiting for $\{N_A, N_B, M\}_{pk(K_A)}$

Bibliography

- John Clark and Jeremy Jacob: A survey of authentication protocol literature, 1997. `http://www.cs.york.ac.uk/~jac/PublishedPapers/reviewV1_1997.pdf`
- Paul Syverson. A Taxonomy of Replay Attacks. CSFW 1994.

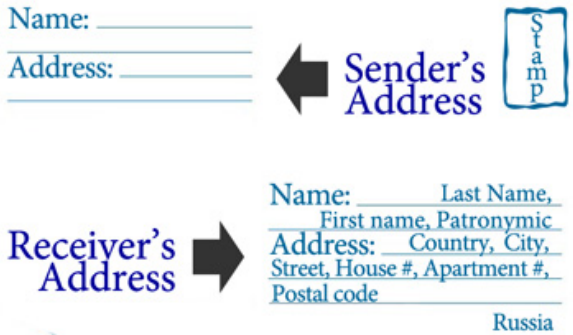
Formal techniques can help

Protocols can be specified by using process calculi

- Process calculi are mathematically rigorous languages with well defined semantics, for formally modelling concurrent and distributed systems
- Protocol agents are modelled as models of processes
- Protocols evolution is described in terms of transitions systems, graphs whose states are process calculi terms
- BUT
- Transition systems are usually huge and their exploration can be computationally demanding
- Static techniques (CFA, TS, AI) provide approximate answers just looking at the system description.

Setting the scene

- **Control Flow Analysis** (CFA) have been applied to cryptographic protocols to detect possible flaws
- CFA soundly over-approximates the behaviour of protocols described in the process algebra LySA; in particular, it tracks message flow for protocol and attacker
- The CFA addresses message authenticity, in the presence of a Dolev-Yao attacker: it verifies that a message encrypted by principal A (origin) and intended for B (destination) does indeed come from A and reaches B only



The Nature of Approximation

Block 1

If no violation of the message authentication property is detected, then no violation will ever arise at run time

Block 2

The existence of violations at static time does not necessarily imply their existence at run time

- BUT, the existence of static violations is a warning bell and should be further investigated

Approach

- Define a protocol (using a process calculus)
- Define a Dolev-Yao style attacker
- Track message flow for protocol and attacker using control flow analysis
- If messages end up in a wrong place then there may be a problem
  - Attacker can alter message flow arbitrarily
  - Focus on encryption and decryption

Quindi quello che facciamo è definire il protocollo usando una LySa, definire lo stile dell'attaccante, questo l'abbiamo già definito è di tipo Dolev-Yao poi vado a tracciare il flusso dei messaggi attraverso l'analisi. L'analisi che abbiamo cominciato a delineare la volta scorsa sostanzialmente è una variazione rispetto a quella che abbiamo visto per il pi-calcolo. Viene complicata dal fatto che LySa come SPI ha una ulteriore categoria sintattica che è quella dei termini che possono essere strutturati rispetto al pi calcolo monadico che abbiamo visto e quindi ho bisogno di analizzare in qualche modo non solo le variabili ma anche termini complessi che al loro interno possono avere più variabili, quindi sostanzialmente la mia analisi mi dirà quali sono i termini che posso avere una volta che ho coperto tutti i buchi, cioè tutte le variabili in tutti i modi possibili, legati appunto ai possibili legami delle variabili interne. Poi andrò a vedere come al solito che cosa passa dalla rete, qui abbiamo fatto l'assunzione in LySa come abbiamo detto l'altra volta che non stiamo a mettere i canali è come se ce ne fosse uno solo, quindi ancora abbiamo rho ancora abbiamo k ma in particolare, k non è più legata a un nome di canale, ma è un k generico, un k globale se volete. A queste due componenti più appunto la parte che riguarda l'analisi dei termini, l'altra variazione rispetto al pi-calcolo è il fatto che esiste una componente di errore psi che mi dice appunto quali sono le coppie di mismatch che sono quelle che dicevamo ci mettono in allarme. Vedremo anche in che modo il nemico entra a far parte del quadro.

Making Narrations Precise

- Typically, protocols are described by narrations and a textual description but details may be imprecise
- We make systematic translations of the protocol narrations into a process calculus called LySA
- LySA is inspired by the Spi-calculus but processes:
  - communicate through a global network
  - match values on input and decryption
  - use symmetric key cryptography

la considerazione che facevamo la volta scorsa sui protocolli è quella che ci porta a scrivere delle specifiche piuttosto precise in LySa dei protocolli tipici, proprio perché abbiamo sempre detto che la notazione informale a volte mette dei dettagli che non sono così secondari e soprattutto lascia un pochino di libertà a chi implementa magari non rendendo esplicite delle ipotesi che invece sono fondamentali, questo porta poi a implementazioni fallaci, quindi alla possibilità di offrire il fianco ad attacchi. LySa, abbiamo detto, è ispirata alla spi calculus, quindi ha primitive di encryption e decryption e comunica attraverso un canale globale e poi la caratteristica particolare che serve un pochino anche ai fini dell'analisi è il fatto che il pattern matching che abbiamo visto anche in spi calculus o nel pi calculus è esterno: tipo x uguale y allora continuo, viene invece incorporato nelle operazioni di input e decryption in modo che contestualmente un processo riceve una tupla e la accetta solo se la tupla comincia con la parte di informazioni che io mi aspetto che la tupla contenga e negli altri casi la butto via, quindi questo ha che fare col fatto che il nemico può mandare i messaggi di ogni genere, ma alla fine gli unici che possono essere accettati in ricezione sono quelli costruiti come il protocollo detta a chi riceve.

questa è infatti la sintassi e lo stesso ragionamento del pattern matching è valido anche per la decryption, l'unica forzatura rispetto a come vengono scritti i protocolli è quello che dobbiamo riarrangiare all'interno delle tuple sia di output che di encryption in maniera tale che le informazioni di conferma vengono date nella parte sinistra, invece, le informazioni che attendo, le attendo nelle variabili che stanno nella parte a destra, poi per il resto la struttura del processo è tipo quella che abbiamo visto per il pi calculus, l'unica cosa, le espressioni naturalmente possono essere composte composite e quindi anche i messaggi sono poliadrici, quindi io posso mandare più espressioni nello stesso output.

LySa Syntax

Expressions

$E ::=$	n	name ($n \in \mathcal{N}$)
	x	variable ($x \in \mathcal{X}$)
	$\{E_1, \cdots, E_k\}_{E_0}$	encryption

Processes

$P ::=$	$\langle E_1, \cdots, E_k \rangle . P$	output
	$(E_1, \cdots, E_j; x_{j+1}, \cdots, x_k) . P$	input (with matching)
	decrypt E as $\{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}$ in P	decryption (with matching)
	$P_1 \mid P_2$	parallel composition
	$(\nu\, n) P$	introduce new name n
	$! P$	replication
	0	terminated process

Quindi questo è un esempio di pattern matching che abbiamo visto l'altra volta, quindi io accetto questa tupla proprio perché comincia per a così come mi aspetto e eccetto il risultato delle encryption e della decryption perché mi aspetto che la prima parte contenga b e poi naturalmente sulla seconda parte del pattern matching vado a fare il binding delle variabili, quindi faccio esattamente tutto come prima solo che collasso in una operazione sola l'input e la decryption con i pattern matching.

LySA features: decryptions on the fly

- No channels: communication is on a single global network.
- Decryptions are embedded inside inputs and performed on the fly when receiving the corresponding outputs. The output

$$\langle A, N_A, \{B, K_{AB}\}_{K_{AS}} \rangle$$

matches the input, including the embedded encryption

$$(A, x_N, \{B, x_K\}_{K_{AS}})$$

and the variables x_N and x_K are bound to N_A and to K_{AB}

LySA features: message authentication

To track message origin and destination, encryption terms and pattern terms are labelled:

$$\{E_1, \cdots, E_k\}^{\ell}_{E_0}[\text{dest } \mathcal{L}] \quad \text{and as} \quad \{M_1, \cdots, M_k\}^{\ell'}_{E'_0}[\text{orig } \mathcal{L}']$$

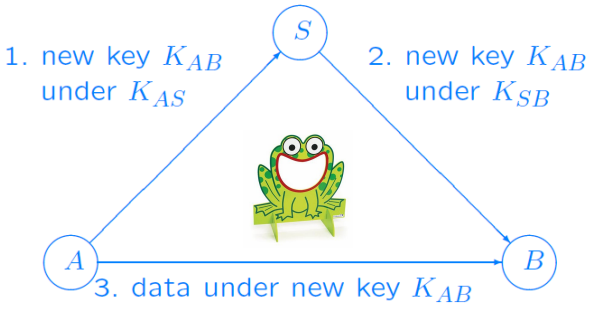
per tracciare invece la proprietà che abbiamo in questo momento sotto osservazione vado a etichettare le encryption e le decryption e questa è una notazione che è fatta ai fini dell'analisi, quindi non sto mettendo nei messaggi reali che faccio scambiare ai partecipanti queste etichette, le metto a uso e consumo delle analisi. Nulla vieta di fare implementazione in cui si aggiungono questi componenti ai messaggi che si scambiano però questo procura un aggravio nella pesantezza proprio dei messaggi. Questa è una proprietà particolare, ma questa è una cosa che riaccade nel senso che se vi ricordate gli avevo fatto vedere, ma veramente molto in velocità gli attacchi di tipo nei protocolli, ovvero molti attaccanti ne approfittano del fatto che chi riceve, non è che riceve la tupla così come la vediamo e la immaginiamo in termini astratti, ma semplicemente riceve una sequenza di bit e sostanzialmente va a fare il parsing guidato dalla forma del messaggio che si aspetta di ottenere e questo è un po rischioso perché può essere che il nemico riesca a fare un reply attack con un messaggio utilizzato precedentemente, che ha la stessa forma, magari anche cifrato con la stessa chiave, ma all'interno invece di esserci una chiave c'è un nonce, e quindi l' ignaro ricevente del messaggio può erroneamente accettare la chiave di sessione, quindi da li in poi esporrà i suoi segreti al nemico. Questo è possibile, appunto, perché il parsing è guidato da quello che uno sa ma non c'è un modo per verificarlo, l'unico modo sarebbe taggare, etichettare, diciamo le componenti del messaggio e su quello fare pattern matching, quindi io accetto la seconda componente di una decryption come chiave di sessione solo se arriva con il tag chiave, se invece arriva con il tag nonce mi accorgo che quello è un replay attack e quindi la scarto.

Encoding a Protocol in LySA

Example

A key exchange inspired protocol by the Wide Mouthed Frog

$$
\begin{array}{lll}
1. & A \to S: & A, B, \{K_{AB}\}_{K_{AS}} \\
2. & S \to B: & A, \{K_{AB}\}_{K_{BS}} \\
3. & A \to B: & \{m_1, \cdots, m_k\}_{K_{AB}}
\end{array}
$$



Encoding a protocol in LySA

Example (Wide Mouthed Frog)

1. $A \to S : \quad A, B, \{K_{AB}\}_{K_{AS}}$
2. $S \to B : \quad A, \{K_{AB}\}_{K_{BS}}$
3. $A \to B : \quad \{m_1, \cdots, m_k\}_{K_{AB}}$

$$(\nu\, K_{AB})\langle \qquad A, B, \{K_{AB}\}_{K_{AS}}\rangle.$$

Encoding a protocol in LySA

Example (Wide Mouthed Frog)

1. $A \to S : \quad A, B, \{K_{AB}\}_{K_{AS}}$
2. $S \to B : \quad A, \{K_{AB}\}_{K_{BS}}$
3. $A \to B : \quad \{m_1, \cdots, m_k\}_{K_{AB}}$

$$(\nu\, K_{AB})\langle A, S, A, B, \{K_{AB}\}_{K_{AS}}\rangle.$$
$$|$$

Encoding a protocol in LySA

Example (Wide Mouthed Frog)

1. $A \to S : \quad A, B, \{K_{AB}\}_{K_{AS}}$
2. $S \to B : \quad A, \{K_{AB}\}_{K_{BS}}$
3. $A \to B : \quad \{m_1, \cdots, m_k\}_{K_{AB}}$

$$(\nu\, K_{AB})\langle A, S, A, B, \{K_{AB}\}_{K_{AS}}\rangle.$$
$$|$$
$$(A, S, A; x_B, x)$$

Encoding a protocol in LySA

Example (Wide Mouthed Frog)

1. $A \to S : \quad A, B, \{K_{AB}\}_{K_{AS}}$
2. $S \to B : \quad A, \{K_{AB}\}_{K_{BS}}$
3. $A \to B : \quad \{m_1, \cdots, m_k\}_{K_{AB}}$

$$(\nu\, K_{AB})\langle A, S, A, B, \{K_{AB}\}_{K_{AS}}\rangle.$$
$$\mid$$
$$(A, S, A; x_B, x).\text{decrypt } x \text{ as } \{; x^K\}_{K_{AS}} \text{ in } \ldots$$

Adding annotations

Example (Wide Mouthed Frog)

1. $A \to S : \quad A, B, \{K_{AB}\}_{K_{AS}}$
2. $S \to B : \quad A, \{K_{AB}\}_{K_{BS}}$
3. $A \to B : \quad \{m_1, \cdots, m_k\}_{K_{AB}}$

$$(\nu\, K_{AB})\langle A, S, A, B, \{K_{AB}\}^{A}_{K_{AS}}\rangle.$$

$$\mid$$

$$(A, S, A; x_B, x).\text{decrypt } x \text{ as } \{; x^{K}\}^{S}_{K_{AS}} \text{ in } \ldots$$

Adding annotations (cont.)

Example (Wide Mouthed Frog)

1. $A \to S : \quad A, B, \{K_{AB}\}_{K_{AS}}$
2. $S \to B : \quad A, \{K_{AB}\}_{K_{BS}}$
3. $A \to B : \quad \{m_1, \cdots, m_k\}_{K_{AB}}$

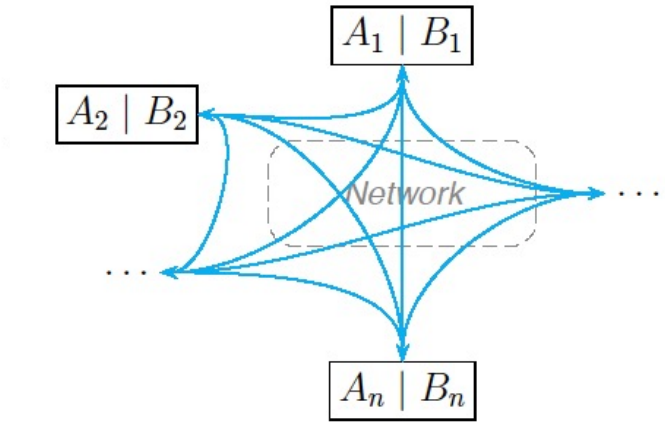
$$(\nu\, K_{AB})\langle A, S, A, B, \{K_{AB}\}^{A}_{K_{AS}}[\text{dest } S]\rangle.$$

$$\mid$$

$$(A, S, A; x_B, x).\text{decrypt } x \text{ as } \{; x^{K}\}^{S}_{K_{AS}}[\text{orig } A] \text{ in } \ldots$$

Protocol encoding: multiple instances

Note that protocols are played by several principals able to play both the initiator role A_i and the responder one B_i, plus a server S, if any



Protocol encoding: multiple instances (2)

There are **multiple instances** of agents playing the roles of initiators, responders and servers. They run in parallel.

0. $(\nu^n_{i-1} K^A_i)(\nu^n_{j-1} K^B_j)$
1. $|^n_{i-1} \; |^n_{\substack{j-1 \\ j \neq i}} \; !(\nu K_{ij})$
   $\langle I_i, A, I_{-1}, S, I_i, A, \{I_j, B, K_{ij}\}^{A_i}_{K^A_i}$ [dest S]$\rangle$.
3. $(\nu m_{1ij}) \cdots (\nu m_{kij})$
   $\langle I_i, A, I_j, B, \{m_{1ij}, \ldots, m_{kij}\}^{A_i}_{K_{ij}}$ [dest B_j]$\rangle$

$2'$. $| \; |^n_{j-1} \; !(I_{-1}, S, I_j, B; y_j)$.
$2''$. $|^n_{i-1}$ decrypt y_j as $\{I_i, A; y^K_{ij}\}^{B_j}_{K^B_j}$ [orig S] in
$3'$. $(I_i, A, I_j, B; z_{ij})$.
$3''$. decrypt z_{ij} as $\{; z^{m_1}_{ij}, \ldots, z^{m_k}_{ij}\}^{B_j}_{y^K_{ij}}$ [orig A_i] in 0

$1'$. $| \; |^n_{i-0} \; !(I_i, A, I_{-1}, S, I_i, A; x_i)$.
$1''$. $|^n_{j-0}$ decrypt x_i as $\{I_j, B; x^K_{ij}\}^S_{K^A_i}$ [orig A_i] in
2. $\langle I_{-1}, S, I_j, B, \{I_i, A, x^K_{ij}\}^S_{K^B_j}$[dest B_j]$\rangle$

LySA the analysis

e naturalmente, come dicevamo, abbiamo più possibili protocolli in contemporanea dove lo stesso agente può giocare più ruoli come iniziatore e responder e può anche in teoria, far eseguire più protocolli, quindi naturalmente dopo l' input ci sarà la decryption in questo caso e a questo punto andiamo ad arricchire la specifica con le etichette che abbiamo detto per tracciare la proprietà, quindi mettiamo che la prima encryption viene fatta da a e ha come destinazione il server e corrispondentemente il server andrà a fare la decryption di qualcosa che deve arrivare da a.

Our *Control Flow analysis* computes an (over-)approximation to

- the messages on the networks: κ
- the values of the variables: ρ
  For instance:

$$\begin{array}{ll} \langle A, S, A, B, \{K_{AB}\}^{A}_{K_{AS}}[\text{dest } S]\rangle & \in \kappa \\ \{K_{AB}\}^{A}_{K_A}[\text{dest } S] & \in \rho(x) \\ K_{AB} & \in \rho(x^K) \end{array}$$

Che cosa fa la CFA? In soldoni come dicevamo ha ancora una componente che mi racconta che cosa passa sulla rete, a questo punto sull'unico canale possibile, quello globale, e quindi k e abbiamo ancora una componente rho che mi dice che cosa può essere legato alle variabili, quindi pensando all' esempio di prima possiamo dire che tutta la tupla passerà sul canale globale, quindi apparterrà a k e poi x verrà legata all'encryption e in particolare una volta fatta la decryption x con k conterrà la nuova chiave di sessione e poi, come dicevamo, abbiamo una componente dell'analisi che mi segna i possibili mismatch, quindi se io trovo che nella mia analisi ho qualcosa che esce e che è stato creato da qualcuno per qualcun altro e questo non è verificato almeno a vedere dalla specifica e dall'analisi allora io metto la coppia di mismatch all'interno di questa famosa componente di errore psi che mi dice sostanzialmente che qualcosa è codificato in a' viene decrittato ad s dove invece l'origine era un'altra oppure viceversa.

LySA: the analysis (cont.)

- the messages on the networks: $\boldsymbol{\kappa}$
- the values of the variables: $\boldsymbol{\rho}$
- the possible mismatches of origin and destination: $\boldsymbol{\psi}$
  E.g. if also $\langle A \to S, A, B, \{K\}_{K_A}^{A'}[\text{dest } S]\rangle \in \kappa$ then

$$(A', S) \in \boldsymbol{\psi}$$

something encrypted at A' may be decrypted at S, where the expected origin was A

Semantics

- Standard reduction semantics $P \to P'$
- The standard semantics ignores annotations

- We can also make a reference monitor semantics $P \to_{\mathrm{RM}} P'$
- The reference monitor gets stuck when annotations are violated

- The reference monitor aborts the execution of P

  whenever $\quad P \to^* Q \to Q'$

  but $\quad P \to^*_{\mathrm{RM}} Q \not\to_{\mathrm{RM}} Q'$

Communication Rule

(Com)

$$\frac{[\![E_1]\!] = [\![E_1']\!]}{\langle E_1, E_2\rangle.P \mid (E_1'; x_2).Q \rightarrow P \mid Q[E_2/x_2]}$$

$$(\nu\, K_{AS})(\langle A, S, A, B, \{K_{AB}\}_{K_{AS}}\rangle.\, A' \mid (A, S, A;\, x_B, x).\, \text{decrypt } x \text{ as } \{; x^K\}_{K_{AS}} \text{ in } B') \rightarrow (\nu\, K_{AS})(A' \mid \text{decrypt } \{K_{AB}\}_{K_{AS}} \text{ as } \{; x^K\}_{K_{AS}} \text{ in } B')$$

Decryption Rule

(Dec)

$$\frac{[\![E_0]\!] = [\![E_0']\!] \wedge [\![E_1]\!] = [\![E_1']\!] \wedge (cond)}{\text{decrypt } \{E_1, E_2\}^{\ell}_{E_0} [\text{dest } \mathcal{L}] \text{ as } \{E_1'; x_2\}^{\ell'}_{E_0'} [\text{orig } \mathcal{L}'] \text{ in } P \to P \mid Q[E_2/x_2]}$$

$$(\nu\, K_{AS})(\langle A, S, A, B, \{K_{AB}\}^{A}_{K_{AS}}[\text{dest } S]\rangle.\, A' \mid (A, S, A;\, x_B, x).\, \text{decrypt } x \text{ as } \{; x^K\}^{S}_{K_{AS}}[\text{orig } S] \text{ in } B')$$

$$\to$$

$$(\nu\, K_{AS})(A' \mid \text{decrypt } \{K_{AB}\}^{A}_{K_{AS}}[\text{dest } S] \text{ as } \{; x^K\}^{S}_{K_{AS}}[\text{orig } A] \text{ in } B')$$

$$\to$$

$$(\nu\, K_{AS})A' \mid B'[K_{AB}/x^K]$$

$$(cond) \quad \text{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) = (\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L})$$

The Analysis

We make a Control Flow Analysis that calculates (an over-approximation) to

- the messages on the network $\kappa \in \mathcal{P}(\mathcal{V}^*)$
- the values of the variables $\rho : \mathcal{X} \to \mathcal{P}(\mathcal{V})$

where $\mathcal{V}$ is the set of values (variable-free terms).

For example:

$$\begin{array}{ll} \langle A, S, A, B, \{K_{AB}\}^{A}_{K_{AS}}[\text{dest } S]\rangle & \in \kappa \\ \{K_{AB}\}^{A}_{K_A}[\text{dest } S] & \in \rho(x) \\ K_{AB} & \in \rho(x^K) \end{array}$$

The Error Component

The error component ψ collects pairs of crypto-points where the assertions in annotations may be violated. For example,

$$(A, B) \in \psi$$

reads

> "Something encrypted at A may *unintentionally* be decrypted at B."

The Analysis

The analysis is specified as a Flow Logic with judgements

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

Gli altri elementi dell'analisi sono come sempre i judgment e quello che succede è che io avrò un jugement generale per i processi che mi dice che una certa estimate è valida rispetto alle solite clausole che adesso vedremo come sono fatte nel caso di LySa e rispetto all' analisi che abbiamo visto in PI Calculus qui ho anche una componente psi che mi serve appunto per mantenermi le coppie di eventuali mismatch. Come dicevamo prima però, la sintassi di LySa è un pò complicata dal fatto che esiste un'altra categoria sintattica che è quella delle espressioni, quindi avrò bisogno anche di analizzare i termini, per questo ci sono degli speciali sotto judgment in cui dico che rho è valido per E : theta, vuol dire che i valori composti che possono essere legati ad E li trovo nella componente theta.

and auxiliary judgements for terms:

$$\rho \models E : \vartheta$$

where $\vartheta \in \mathcal{P}(\mathcal{V})$ is the values that E may evaluate to

Judgements for terms

$$\frac{\lfloor n \rfloor \in \vartheta}{\rho \models n : \vartheta} \qquad \frac{\rho(\lfloor x \rfloor) \subseteq \vartheta}{\rho \models x : \vartheta}$$

encryption

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

If, e.g., $E = \{A, x^K\}^{S}_{K_B}[\text{dest } B]$ and $K \in \rho(x^K)$, then

$$\{A, K\}^{S}_{K_B}[\text{dest } B] \in \vartheta$$

Judgement for Output

(of binary terms)

binary output

$$\frac{\begin{array}{c}\rho \models E_1 : \vartheta_1 \wedge \rho \models E_2 : \vartheta_2 \\ \forall V_0, V_1 : V_0 \in \vartheta_0 \wedge V_1 \in \vartheta_1 \Rightarrow \langle V_1, V_2 \rangle \in \vartheta \wedge \\ (\rho, \kappa) \models P : \psi\end{array}}{(\rho, \kappa) \models \langle E_1, E_2 \rangle . P : \psi}$$

- Analyse the expressions E_1, E_2
- ϑ includes the pairs of the values possibly bound to E_1, E_2
- Analyse the continuation

Judgement for Input

questo invece è il judgement per l'input che mi va a capire che cosa può essere legato sostanzialmente alla variabile da legare, cioè x. Quindi lui si aspetta di avere di andare a controllare tutte le tuple che cominciano per e1 che stanno in giro, che cosa vuol dire che stanno in giro. Intanto io devo essere sicuro a cosa corrisponde e1 a run time, quindi vado a vedere in theta1 tutti i possibili valori, anche compositi, che uno può assumere, a questo punto vado a vedere tutte le coppie in k, che hanno la forma che io mi aspetto, cioè io mi aspetto una coppia, quindi vado a prendere tutte le coppie che passano in rete e vado a vedere tutte le coppie che cominciano per v1, se v1 appartiene a theta1, cioè se v1 può andare a run time a corrispondere a e1 allora vuol dire che la mia analisi è fatta bene quando avrà previsto che x possa essere legata alla variabile. Quindi come nel PI calculus, quando vado a fare l' input vado a vedere sostanzialmente che cosa c'è in k e che cosa può matchare, nel PI calcolo l'unica cosa che andava a matchare era sostanzialmente il nome del canale, qui il nome del canale non ce l'ho più, vado a matchare però la prima parte del messaggio laddove c'è una parte del messaggio da da far corrispondere. Per ogni input, vado a vedere tutte le tuple che possono fare matching e se ne trovo, lego in tutti i modi possibili la variabile x nel modo adeguato.

(of binary terms)

binary input

$$\frac{\begin{array}{ll}\rho \models E_1 : \vartheta_1 \wedge & \text{evaluate subterms}\\ \forall \langle V_1, V_2 \rangle \in \kappa : & \text{for all output tuples}\\ \quad V_1 \text{ E } \vartheta_1 \Rightarrow & \text{if values match}\\ \quad\quad V_2 \in \rho(x_2) \wedge & x_2 \text{ has the value } V_2\\ (\rho, \kappa) \models P : \psi & \text{analyse the continuation}\end{array}}{(\rho, \kappa) \models (E_1; x_2).P : \psi}$$

Judgement for Decryption

lo stesso principio lo posso utilizzare per la decryption, quindi vi rinnovo l'invito a vedere nella decryption una sorta di input. Che cosa succede nel in questa sorta di input? Di nuovo ho da far corrispondere il messaggio che devo decifrare con quello che io suppongo che sia. Quindi per farlo io che cosa faccio? Vado a prendere l'espressione e che devo decifrare, vado a vedere che cosa mi restituisce l'analisi al posto di e e se l'analisi mi restituisce un oggetto che è nella forma che io spero, cioè per esempio in questo caso, una coppia codificata con una chiave E0 se questo è il caso, quindi vado a prendere tutte le coppie concrete che hanno l'arità corretta e vado a vedere se corrispondentemente la chiave concreta può essere una chiave che a run time trova il posto di E0 e se il primo elemento della coppia che codifico v1, fa parte delle possibili istanziazioni di e1 a runtime, se questo succede vuol dire che la decryption può essere effettuata, almeno nella semantica standard, e quindi può essere che v2 possa essere legato alla variabile x2. Naturalmente devo ancora assicurarmi che l'analisi valga per la continuazione, in più in blu ho il controllo delle annotazioni che devo inserire per vedere se la mia proprietà sul messaggio origine destinazione è rispettata. Se non lo fosse, quindi se io mi trovo nell'analisi di e sostanzialmente qualcosa che non va a matchare quindi o da una parte dall'altra, quindi, o l' che non appartiene a l o l che non appartiene a l', allora io quella coppia me la vado a isolare e mettere dentro la componente psi oppure se la vedete in termini di controllo io vado a controllare che se quella è il caso, la mia analisi deve aver indovinato che l e l' non possono corrispondersi. Quindi questo è un po l'idea, poi sostanzialmente, io vado tutte le volte a vedere che cosa succede staticamente.

(of binary terms)

binary decryption

$$\frac{\begin{array}{ll}
\rho \models E : \vartheta \wedge & \text{evaluate terms} \\
\rho \models E_0 : \vartheta_0 \wedge \rho \models E_1 : \vartheta_1 \wedge & \text{and subterms} \\
\forall \{V_1, V_2\}^{\ell}_{V_0}[\text{dest } \mathcal{L}] \in \vartheta : & \text{for all encrypted terms} \\
\quad V_0 \,\mathsf{E}\, \vartheta_0 \wedge V_1 \,\mathsf{E}\, \vartheta_1 \Rightarrow & \text{if values match} \\
\quad\quad V_2 \in \rho(x_2) \wedge & x_2 \text{ has the value } V_2 \\
\quad\quad \ell' \notin \mathcal{L} \vee \ell \notin \mathcal{L}' \Rightarrow (\ell, \ell') \in \psi \wedge & \text{check annotations} \\
\quad\quad (\rho, \kappa) \models P : \psi & \text{analyse the continuation}
\end{array}}{(\rho, \kappa) \models \text{decrypt } E \text{ as } \{E_1;\, x_2\}^{\ell'}_{E_0}\, [\text{orig } \mathcal{L}'] \text{ in } P : \psi}$$

Formal results

- **Correctness**: the analysis is sound w.r.t. the semantics

Theorem

(Subject Reduction)

If $(\rho, \kappa) \models P : \psi$ *and* $P \to Q$ *then also* $(\rho, \kappa) \models Q : \psi$

Theorem

($\psi = \emptyset$ *means we're happy)*

If $(\rho, \kappa) \models P : \emptyset$ *then the reference monitor cannot abort the execution of* P*.*

- **Existence of solutions**: valid estimates are a Moore family

LySA: The Dolev-Yao attacker

The attacker is not specified at the calculus level. It is specified at analysis level (using ρ and κ) with a logic formula.

- The attacker knowledge is kept in a special variable $z_\bullet$
- The attacker has a special crypto-point called $\ell_\bullet$

E.g., its intercepting and injecting capabilities are written as

$$\forall \langle V_1, \cdots, V_k \rangle \in \kappa : \wedge_{i=1}^{k} V_i \in \rho(z_\bullet)$$

$$\forall V_1, \cdots, V_k : \wedge_{i=1}^{k} V_i \in \rho(z_\bullet) \Rightarrow \langle V_1, \cdots, V_k \rangle \in \kappa$$

Validating Authentication

Definition

P guarantees dynamic authentication if $P \mid Q$ cannot abort regardless of the choice of the attacker Q.

Definition

P guarantees static authentication if $(\rho, \kappa) \models P : \emptyset$ and $(\rho, \kappa, \emptyset)$ satisfies the formula describing the attacker.

Theorem

If P guarantees static authentication then P guarantees dynamic authentication.

Implementation

- Transform the analysis into (an extension of) Horn clauses.
- Calculate the solution using the Succinct Solver.
- Main challenge:
  - The analysis is specified using the infinite universe of terms.
  - Use an encoding of terms in tree grammars where terms are represented as a finite number of production rules.
- Runs in polynomial time in the size of the process P.

Ora, dal punto di vista di come è fatta e implementata ,queste clausole che abbiamo visto, si possonotrasformare in clausole di Horn e calcolare con apposito risolvitore e si ha che tutto può essere eseguito in polynomial time ma andiamo a vedere il nostro esempio di prima. ora

Example Revisited

Example

$$\begin{array}{ll} A \to S: & A, B, \{K_{AB}\}_{K_{AS}} \\ S \to B: & A, \{K_{AB}\}_{K_{BS}} \\ A \to B: & \{m_1, \cdots, m_k\}_{K_{AB}} \end{array}$$

The analysis of n instances of the protocol gives

$$\psi = \{\ (A_i, B_j), (A_i, \ell_\bullet), (\ell_\bullet, B_j) \mid 1 \leq i, j \leq n\}$$

Possiamo vedere che effettivamente se si analizza come abbiamo visto la componente di errore non sarà vuota, ora però vi invito anche a controllare indipendentemente da cosa dice l'analisi, vi prego di far mente locale su quanto abbiamo detto sui processi dei protocolli precedentemente e vi chiedo se secondo voi il protocollo costruito così funziona bene? Quindi adesso considerate solo la narrazione del protocollo. Quindi naturalmente c'è una triangolazione in cui l'informazione sensibile è quella della chiave di sessione e quella sembra protetta, lo è crittograficamente, ma nel contesto del protocollo lo è un po' meno, qualcuno mi sa dire quali problemi potenziali vede? A e B sono inviati in chiaro questo è un problema il fatto che a e B siano inviati in chiaro mi crea sicuramente un problema, perché cosa potrebbe fare il nemico? Potrebbe andare a sostituire A o B spacciandosi quindi per A o B. Se al posto di B ci mette C diciamo che quello che succede è che a quel punto il server prende la chiave e la encripta con la chiave di B. A quel punto A pensa di comunicare con B invece B non è mai stato incluso nel quadro perché tutti i messaggi sono stati fermati prima da C. Uscendo un attimo dall' idea della perfect encryption, potete immaginare che B qui non venga avvertito di questa comunicazione senza averla lui stimolata ,quindi in realtà è anche possibile che quella chiave che riceve, ovviamente con la sua chiave longterm, possa essere semplicemente il replay di una vecchissima run dello stesso protocollo vecchia abbastanza perché il nemico possa aver decriptato la chiave di sessione e quindi a quel punto B potrà ricevere dei messaggi dal nemico pensando che sia A e a sua volta mandarne e quindi esponendo dei segreti a chi non dovrebbe.

An attack

1.	$A \to S:$	$A, B, \{K_{AB}\}_{K_{AS}}$		1.	$A \to S:$	$A, B, \{K_{AB}\}_{K_{AS}}$
2.	$S \to B:$	$A, \{K_{AB}\}_{K_{BS}}$		2.	$S \to M(B):$	$A, \{K_{AB}\}_{K_{BS}}$
3.	$A \to B:$	$\{m_1, \cdots, m_k\}_{K_{AB}}$		$2'$.	$M(S) \to B:$	$C, \{K_{AB}\}_{K_{BS}}$
				3.	$A \to B:$	$\{m_1, \cdots, m_k\}_{K_{AB}}$

E' possibile che il nemico faccia sì che la prima parte del protocollo vada liscia, però intercetta sostanzialmente il messaggio del server e, approfittando del fatto che anche nel secondo messaggio la parte dell'informazione che dice a B chi è che vuole parlare con lui, cioè A, il nemico in realtà può anche su questa intervenire e prendere A e sostituirci C, quindi da lì in poi B penserà di spedire a C invece che spedire ad A. Questo è un'altro possibile attacco, diciamo l'idea è sempre quella di prima. Laddove ci sono informazioni in chiaro, il nemico ci può mettere sempre lo zampino e quindi in termini proprio di protocollo va ad alterare l'accoppiamento che dovrebbe esserci tra chiave e agente con cui la condivido. Tant'è che questo è uno dei principi poi del prudente engeneering dei protocolli che dice che le chiavi invece dovrebbero essere accompagnate e protette insieme alle informazioni che servono, quindi qui ciò che non è protetto è l'associazione tra la chiave e chi la possiede insieme a te, quindi metterlo in chiaro è chiaramente un errore.

Attack (cont.)

1.	$A \to S :$	$A, B, \{K_{AB}\}_{K_{AS}}$	1.	$A \to S :$	$A, B, \{K_{AB}\}_{K_{AS}}$
2.	$S \to B :$	$A, \{K_{AB}\}_{K_{BS}}$	2.	$S \to M(B) :$	$A, \{K_{AB}\}_{K_{BS}}$
3.	$A \to B :$	$\{m_1, \cdots, m_k\}_{K_{AB}}$	$2'$.	$M(S) \to B :$	$C, \{K_{AB}\}_{K_{BS}}$
			3.	$A \to B :$	$\{m_1, \cdots, m_k\}_{K_{AB}}$

The analysis of n instances of the protocol gives:

$$(A_i, B_j) \in \psi$$

Other attacks

$$\begin{array}{ll} A \to M_S: & A, B, \{K\}_{K_A} \\ M_A \to S: & A, B', \{K\}_{K_A} \\ S \to B': & \{A, K\}_{K_{B'}} \\ A \to M_B: & \{m_1 \cdots m_k\}_K \\ M_A \to B': & \{m_1 \cdots m_k\}_K \end{array}$$

$$\begin{array}{ll} A \to M_S: & A, B, \{K\}_{K_A} \\ M_S \to S: & A, M, \{K\}_{K_A} \\ S \to M: & \{A, K\}_{K_M} \\ A \to M_B: & \{m_1 \cdots m_k\}_K \end{array}$$

$$\begin{array}{ll} A \to M_S: & A, B, \{K\}_{K_A} \\ M_S \to S: & A, M, \{K\}_{K_A} \\ S \to M: & \{A, K\}_{K_M} \\ M_A \to S: & A, B, \{K\}_{K_A} \\ S \to B: & \{A, K\}_{K_B} \\ M \to B: & \{m_1 \cdots m_k\}_K \end{array}$$

General Considerations (1)

- The analysis identifies the well-known attacks on the protocols we have considered (Needham-Schroeder, Otway-Rees, Yahalom and Andrew Secure RPC).
- When well-known amendments are performed the analysis reports that there are no attacks
- The analysis discovers an undocumented attack for the Beller-Chang-Jacobi MSR protocol.
- Very few false positives
- Polynomial time validation procedure
- Improve the precision of the analysis

General Considerations (2)

The same approach is valid also for

- other cryptographic features, such as asymmetric cryptography (the calculus has been extended)
- other security properties (by checking other annotations): e.g., on freshness, type flaws
- other calculi: variations of LySA

Che cosa è stato fatto con questa analisi a suo tempo? E' stata applicata a un certo numero di protocolli a chiave simmetrica e anche asimmetrico a dire la verità classici quelli che trovate nella survey e è stato trovato appunto quasi sempre una corrispondenza tra le nostre componenti di errore e effettivi attacchi riportati dalla letteratura in alcuni casi ne abbiamo trovati di nuovi, però insomma, sicuramente abbiamo confermato quelli classici e anche che alcuni fix erano effettivamente necessari, diciamo e chiaramente quindi questo è una tipo di approccio che può essere poi esteso ad altre caratteristiche crittografiche, è stato esteso poi il calcolo anche alla parte asimmetrica, si può ipotizzare di seguire cambiando il tipo di annotazioni, le proprietà di sicurezza che vogliamo analizzare e naturalmente è comunque un contesto, un framework formale che può essere applicato anche a calcoli diversi o estensioni di questo calcolo qua, quindi questo è un po l'idea che sta dietro all' analisi di LySa per quanto riguarda i protocolli, a questo punto, come trailer della prossima volta, perché non vorrei mettere altra carne al fuoco oggi vi posso raccontare qual è stata l'evoluzione anche dal punto di vista dell'analisi su questo tipo di linguaggio, ne sono state fatte molte variazioni, quindi sono state studiate altri modi, per esempio per catturare problemi di tipo, per catturare problemi di di freschezza e insomma, quindi l'analisi che vi ho mostrato è stata raffinata in più modi, con variazioni sul tema anche rispetto alle proprietà. L'altra evoluzione che ha avuto è stata nella direzione della modellazione, cioè fin qui ci siamo guardati i protocolli, invece l'altra cosa su cui abbiamo cominciato a lavorare è stata invece quella di un'altro scenario, lo scenario dell'IoT e quindi abbiamo a un certo punto esteso la sintassi di LySa in maniera tale da poter modellare non più solo i protocolli, ma anche diciamo sistemi di tipo IoT ovvero abbiamo inserito a nel quadro anche la possibilità di avere sensori e attuatori e abbiamo provato a fare un'analisi anche del loro di comportamento. In quel caso lì l'analisi è stata fatta in maniera diversa, pensando naturalmente a proprietà diverse e in particolare quello che è stato studiato in termini astratti è la composizione dei messaggi che circolano in sistemi di quel genere lì e che sono quelli che stabiliscono sostanzialmente un flusso tra i dati che arrivano dai sensori ai processi che ne controllano la logica e che quindi ipotizzano e poi fanno eseguire quali sono le azioni sugli attuatori più opportune. Abbiamo visto che dal punto di vista della sicurezza una cosa importante è stabilire la bontà dei messaggi che circolano, dei dati che circolano e che sono quelli che poi vanno a incidere sul tipo di azioni che fanno gli attuatori, il che può avere anche conseguenze, quindi se i dati che si analizzano per fare certe azioni che non sono corrette, perché sono non trusted o in qualche modo contaminati, quello che succede è che poi le azioni che si decide di fare poi potrebbero essere anche pericolose e per fare questo non c'è bisogno di sapere qual è il dato esatto che che può circolare sempre in termini di approssimazione, quindi non ho bisogno di sapere quant'è la temperatura, mi serve di sapere la natura delle informazioni e quindi il fatto che queste informazioni possano essere o meno attaccabili.