Control Flow Analysis for process algebras with applications to security

Chiara Bodei[1] and many others

Dip. di Informatica, Università di Pisa, Italy

De Cifris Athesis Seminar
Università degli Studi di Trento, September 10

Outline

1. Background on security protocols
2. CFA and security protocols
3. Possible further directions of investigation
4. Conclusions

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:

- 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)

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

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

Bank transfer protocol (cont.)

Example (Possible fix)

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

Bank transfer protocol (cont.)

Example (Replay attack)

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

- 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

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.

Static vs dynamic techniques

STATIC	*DYNAMIC*
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

CFA applied to cryptographic protocols*

- **Control Flow Analysis** (CFA) soundly over-approximates the behaviour of protocols described in the process algebra LySa
- It tracks message flow for protocol, in the presence of a Dolev-Yao attacker
- The CFA addresses message authenticity: it verifies that a message encrypted by principal A (origin) and intended for B (destination) does indeed come from A and reaches B only
- If messages end up in a wrong place then there may be a problem

* C. Bodei and M. Buchholtz and P. Degano and F. Nielson and H. Riis Nielson (2005) Static Validation of Security Protocols, *Information and Computation* 13(3), pp. 347-390.

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

LySA model

- Typically, protocols are described by narrations and a textual description but details may be imprecise
- Protocol narrations can be made systematically translated into the 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

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

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:

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

A decryption made in ℓ' with required **origin** in [orig $\mathcal{L}'$] matches an encryption in ℓ with required **destination** in [dest $\mathcal{L}$] if

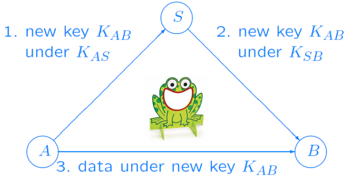
$$\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L}$$

Encoding a Protocol in LySa

Example

A key exchange inspired protocol by the Wide Mouthed Frog

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



Encoding a protocol in LySa

Example (Wide Mouthed Frog)

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

New names are modelled with restriction

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

$$|$$

Encoding a protocol in LySa

Example (Wide Mouthed Frog)

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

Sender and receiver names are put at the beginning of the message

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

$$|$$

Encoding a protocol in LySa

Example (Wide Mouthed Frog)

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

Agents are represented as parallel process (parallel composition is $|$)

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

$$|$$

$$(A, S, A; x_B, x)$$

Encoding a protocol in LySA

Example (Wide Mouthed Frog)

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

Decryption is applied to the value received in x

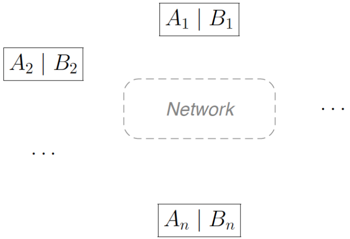
$$(\nu\, K_{AB})\langle A, S, A, B, \{K_{AB}\}_{K_{AS}}\rangle.$$

$$|$$

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

Protocol encoding: multiple instances

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



Adding annotations

Example (Wide Mouthed Frog)

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

Encryptions and decryptions come with labels

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

$$\mid$$

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

Adding annotations (cont.)

Example (Wide Mouthed Frog)

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

Encryptions and decryptions come with annotations on the required destination and origin

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

$$|$$

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

Semantics

- Standard reduction semantics $P \rightarrow P'$
- Standard semantics ignores origin and destination annotations
- We can also make a reference monitor semantics $P \rightarrow_{\mathsf{RM}} P'$
- The reference monitor gets stuck when annotations are violated
- The reference monitor aborts the execution of P

whenever $P \rightarrow^* Q \rightarrow Q'$

but $P \rightarrow^*_{\mathsf{RM}} Q \not\rightarrow_{\mathsf{RM}} Q'$

Communication Rule

(Binary 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

(Binary Dec)

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

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

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

The Analysis

Control Flow Analysis calculates (an over-approximation) to

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

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 possible mismatches of origin and destination: ψ

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

reads

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

E.g. if also $\langle A \rightarrow S, A, B, \{K\}_{K_A}^{A'}[\mathrm{dest}\ S]\rangle \in \kappa$ then

$$(A', S) \in \psi$$

because the expected origin was A

The Analysis

The analysis is specified as a Flow Logic with judgements

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

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

$$\frac{\lfloor n \rfloor \in \vartheta}{\rho \models n : \vartheta}$$

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

encryption

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

For instance, if $E = \{A, x^K\}^{S}_{K_B}[\text{dest } B]$ 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

(of binary terms)

binary input

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

$$\frac{\rho \models E_1 : \vartheta_1 \wedge \forall \langle V_1, V_2 \rangle \in \vartheta : V_1 \text{ E } \vartheta_1 \Rightarrow V_2 \in \rho(x_2) \wedge (\rho, \kappa) \models P : \psi}{(\rho, \kappa) \models (E_1; x).P : \psi}$$

Judgement for Decryption

(of binary terms)

binary decryption

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

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 inputs are labelled $Z_\bullet$
- 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.

Example Revisited

Example

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

The analysis of n instances of the protocol gives

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

If messages end up in a wrong place then there may be a problem

An attack

Investigating because of the warning bell

Example (protocol and protocol 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.	$A \to B :$	$\{m_1, \cdots, m_k\}_{K_{AB}}$

B believes that K_{AB} comes from C

Other attacks

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

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

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

General Considerations

The presented analysis identifies a number of authentication flaws in symmetric key protocols such as Needham-Schroeder, Otway-Rees, Yahalom and Andrew Secure RPC.
The same approach is valid also for

- other cryptographic features, such as asymmetric cryptography (by extending the calculus)
- other security properties (by checking other annotations)
- other calculi

Other protocols and other paradigms

- Multi-factor authentication
- Internet of Things

Considerations on DY attacker*

- It has total control over all channels
  - this is not always realistic
  - attacking a channel has a cost (cheap or in terms of risk)
- It cannot compromise endpoints
  - browser, PC, smartphones
  - nonetheless this can happen (e.g. malware)

* Slides on multi-factor authentication are inspired by the ones of G. Costa

Multi-factor authentication

IDEA: increase the cost by increasing the factors that need to be compromised for carrying out the attack

Typical context: e-banking (a user must be authenticated by her/his bank)

Authentication factors

Authentication Factor: something that identifies who holds the control, e.g. credit card

Three types of AF

- what you know, e.g. a password, or a pin code;
- what you have, e.g. an access card or physical token
- what you are, e.g. a biometric measurement

Authenticator: device used for attesting control over one or more AF, e.g. a sw or hw keyboard

Evidence: generated by an authenticator, an information to demonstrate control over an AF, e.g. ID + ccv

Secure element

- Some authenticators are designed for be resistant against tampering, for example, SIM, Smartcard, OTP generators
- Virtually never this happens if the authenticator is software: the execution platform is untrusted

Attacker model revised

- Device Thief: can steal physical objects at owner
- Authenticator Duplicator: can duplicate an authenticator (except secure elements)
- Shoulder Surfer: can spy on the activity user
- Eavesdropping software: can spy the input of data by the user (e.g. keylogger)
- Social Engineer: can induce a user misinformed to use an authenticator in incorrect way
- Man in the Browser: has control of user browser (if the endpoint is a PC)
- Man in the Mobile: has control of the user smartphone

Stronger attackers are obtained for composition

Authentication with a card reader

Example

$$
\begin{array}{ll}
Alice \rightarrow Bank : & \{usr, h(psw)\}_{K_B} \\
Bank \rightarrow Alice : & \{N_B\}_{K_A} \\
Alice \rightarrow CReader : & N_B, pin \\
CReader \rightarrow Alice : & [N_B]_{K_{BC}} \\
Alice \rightarrow Bank : & [N_B]_{K_{BC}}
\end{array}
$$

A shoulder surfer can compromise knowldege factors, but cannot compromise possesion factors

Attack

Possible ways to attack the factors of possession, with e.g. a social engineer

- DT (subtracts CReader from Alice)
- (deceives Alice who uses CReader badly)
- (create a copy of CReader)
- But CReader is a secure element

Attack

Example

$$
\begin{array}{ll}
Charlie \rightarrow Bank: & \{usr, h(psw)\}_{K_B} \\
Bank \rightarrow Charlie: & \{N_B\}_{K_A} \\
Charlie \rightarrow Alice: & \{N_B\}_{K_A} \\
Alice \rightarrow CReader: & N_B, pin \\
CReader \rightarrow Alice: & [N_B]_{K_{BC}} \\
Alice \rightarrow Charlie: & [N_B]_{K_{BC}} \\
Charlie \rightarrow Bank: & [N_B]_{K_{BC}}
\end{array}
$$

Possible countermeasure: tell Alice

Example

$$
\begin{array}{ll}
Charlie \rightarrow Bank : & \{usr, h(psw)\}_{K_B} \\
Bank \rightarrow Charlie : & \{N_B, Op\}_{K_A} \\
Charlie \rightarrow Alice : & \{N_B, Op\}_{K_A} \\
Alice \ldots. &
\end{array}
$$

Alice can refuse to perform Op

IoT Challenges

Designing & implementing IoT systems is hard

- Smart objects are heterogeneous
  - Many hw, sw, protocols, etc.
- Cyber-physical and dynamic systems
- Security & privacy
  - Highly critical but difficult to achieve
  - Devices (Things) have limited computational capabilities and are battery powered

CFA for Internet of Things*

The process algebra IoT-LySa extends LySa

- Network of nodes
- Sensors & actuators
- Group communication
- Local communication à la Linda

Tracking data analysis to predict

- Interaction among nodes
- How data flow in the network and are manipulated, also related to security issues

* C. Bodei, P. Degano, G-L. Ferrari, L. Galletta. Tracing where IoT data are collected and aggregated. Logical Methods in Computer Science 13(3:5), pp. 1-38, 2017.

Tracking data analysis for security properties

A Control Flow Analysis (CFA) to safely approximate

- Interactions among nodes
- How data spread from sensors to network
- How data are manipulated

Verification based on the analysis results

- Checking whether classified data reach untrusted nodes
- Checking whether data come from untrusted sources
- ...

Conclusions

- Security protocols can achieve properties that cryptographic primitives alone cannot offer, e.g. authentication, secrecy, ...
- Even three lines show how difficult the art of correct design is
- Formal models of protocols and of their properties is required, to provide a mathematically sound basis for reasoning
- However, formal analysis of protocols is non trivial (even assuming perfect cryptography)
- New paradigms come with new security challenges

Thanks

THANK YOU FOR YOUR ATTENTION