Attribute-based Communication meets Security

Michele Loreti joint work with Sara Longhi and Marco Quadrini

University of Camerino

International Workshop on Asynchronous Programming Models
Special edition @ ETAPS 2026
16-17 April 2026, Torino, Italy

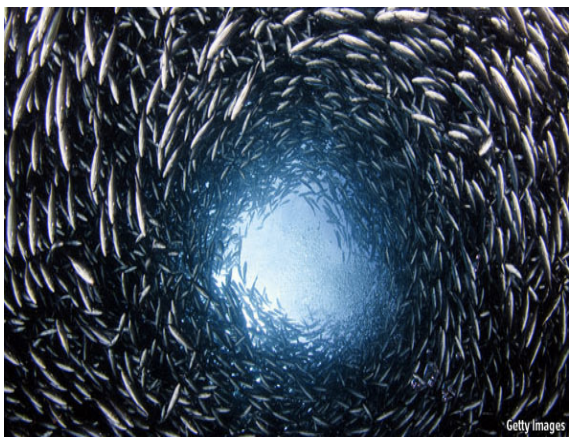
Collective Systems

We are surrounded by examples of collective systems:

Collective Systems

We are surrounded by examples of collective systems:

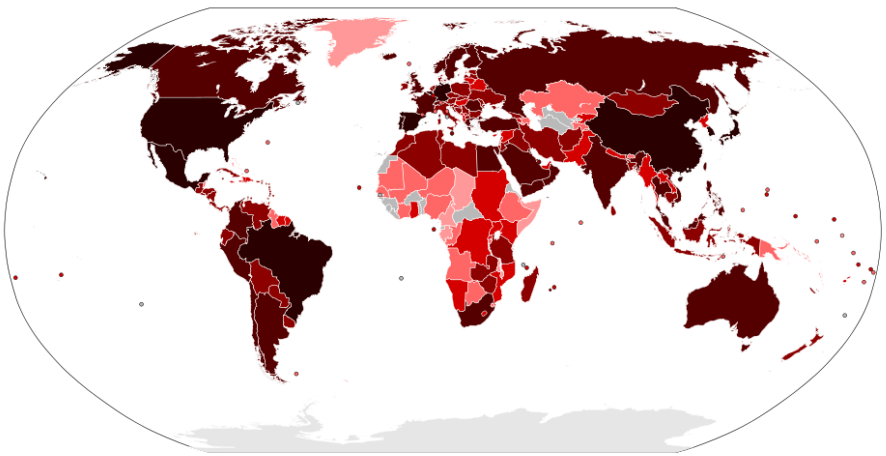
in the natural world ....



Collective Systems

We are surrounded by examples of collective systems:

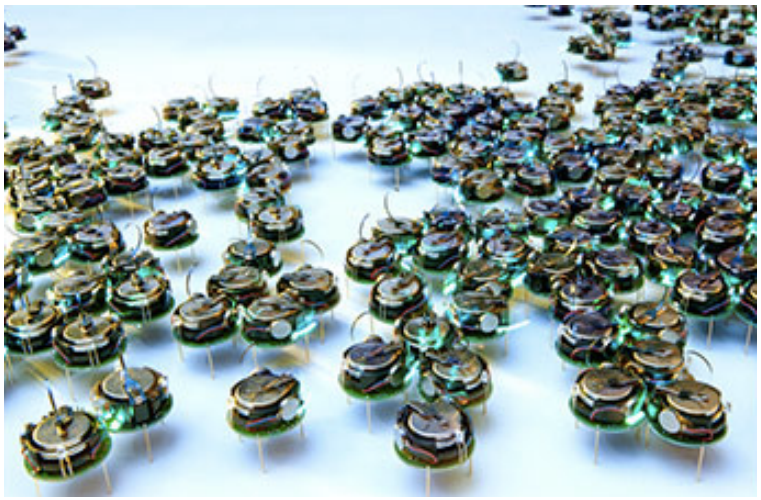
.... and in the man-made world



Collective Systems

We are surrounded by examples of collective systems:

.... and in the man-made world



Interaction Patterns...

These systems exhibit two kinds of interaction patterns:

Interaction Patterns...

These systems exhibit two kinds of interaction patterns:

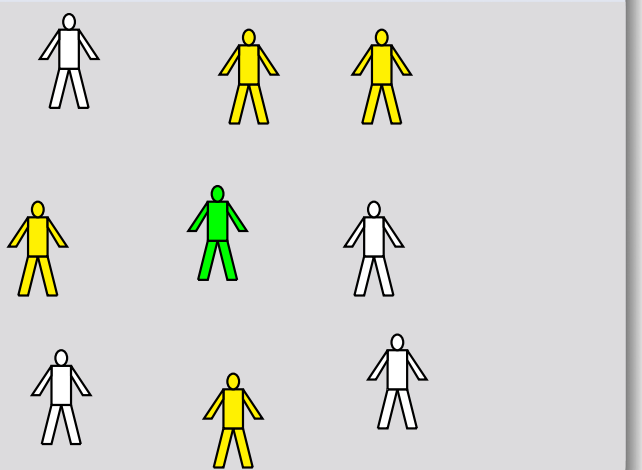
1. **Spreading**: one agent spreads relevant information to a given group of other agents

Interaction Patterns...

These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents

Spreading...

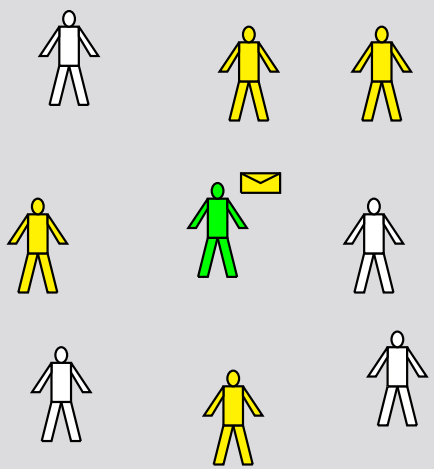


Interaction Patterns...

These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents

Spreading...

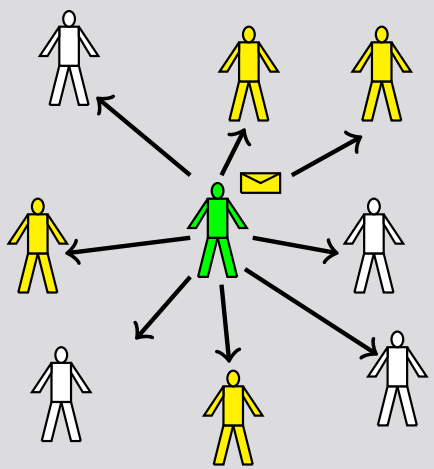


Interaction Patterns...

These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents

Spreading...

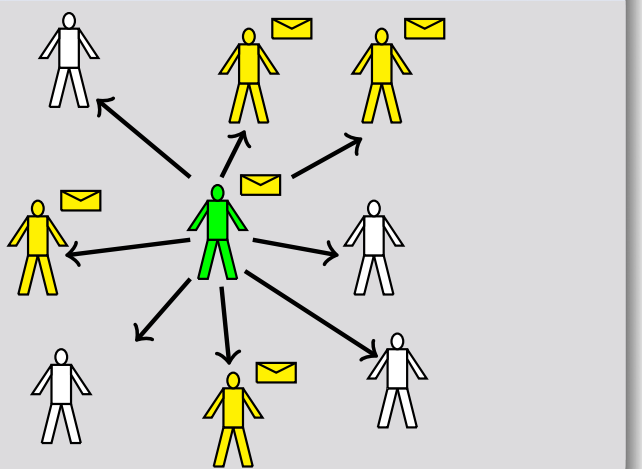


Interaction Patterns...

These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents

Spreading...

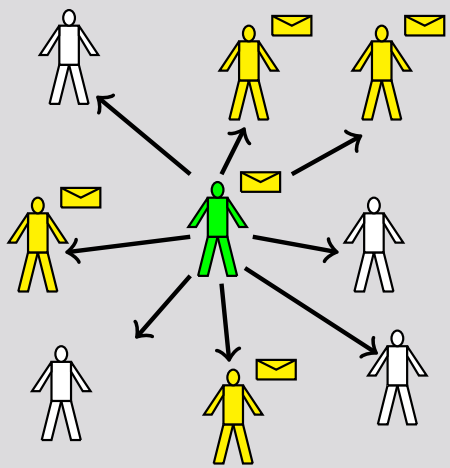


Interaction Patterns...

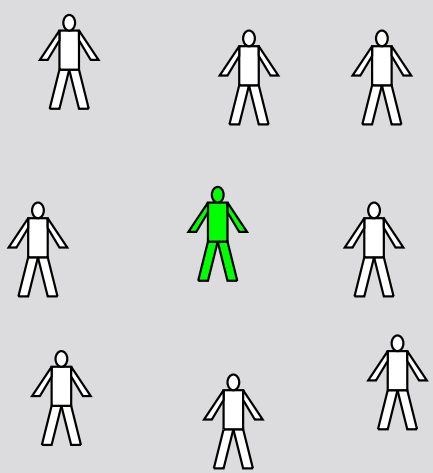
These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents
2. **Collecting**: one agent changes its behaviour according to data collected from one agent belonging to a given group of agents.

Spreading...



Collecting...

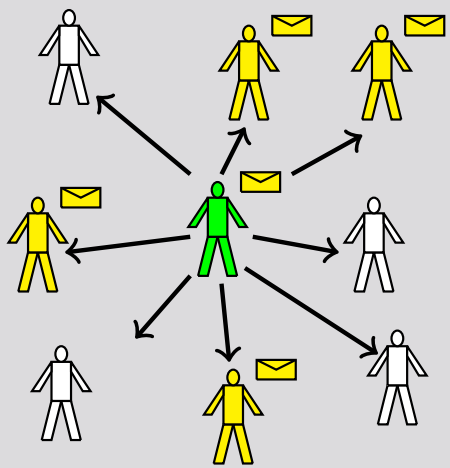


Interaction Patterns...

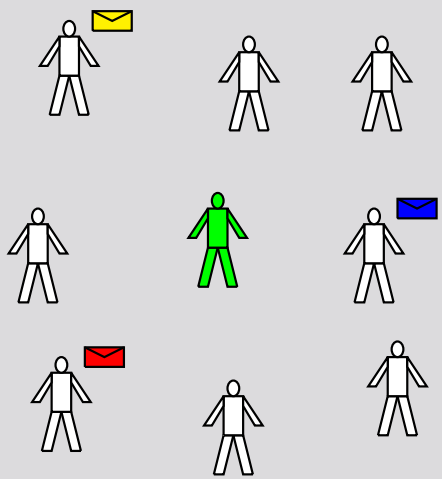
These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents
2. **Collecting**: one agent changes its behaviour according to data collected from one agent belonging to a given group of agents.

Spreading...



Collecting...

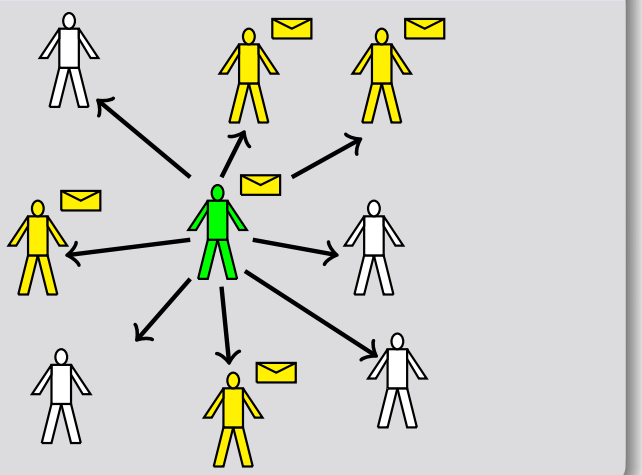


Interaction Patterns...

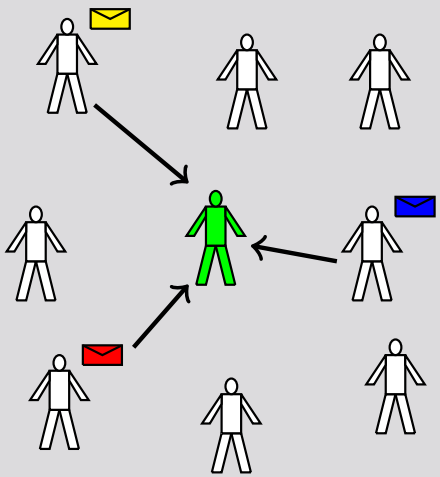
These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents
2. **Collecting**: one agent changes its behaviour according to data collected from one agent belonging to a given group of agents.

Spreading...



Collecting...

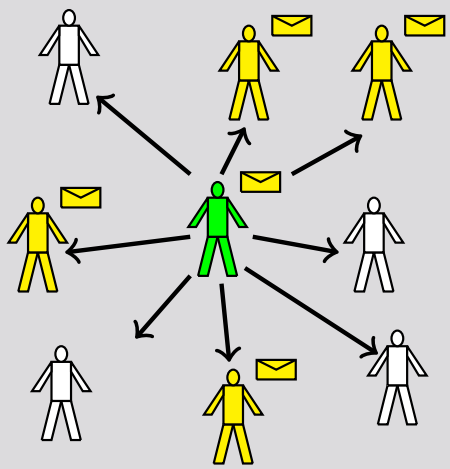


Interaction Patterns...

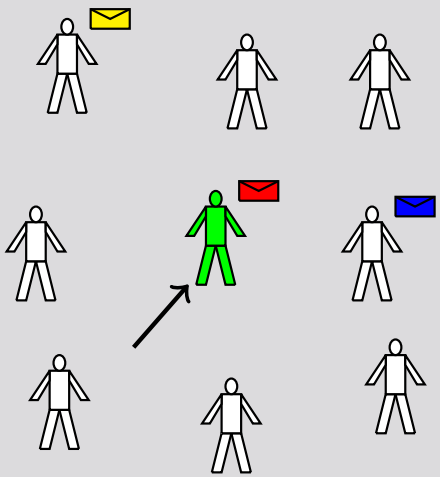
These systems exhibit two kinds of interaction patterns:

1. **Spreading**: one agent spreads relevant information to a given group of other agents
2. **Collecting**: one agent changes its behaviour according to data collected from one agent belonging to a given group of agents.

Spreading...



Collecting...



Attribute-based Communication...

In Attribute-based Communication, all the participants are equipped with a set of attributes.

Attribute-based Communication...

In Attribute-based Communication, all the participants are equipped with a set of attributes.

A message can be sent to all participants satisfying the given predicate π over these attributes.

Attribute-based Communication...

In Attribute-based Communication, all the participants are equipped with a set of attributes.

A message can be sent to all participants satisfying the given predicate π over these attributes.

Receivers filter incoming messages using predicates that specify properties of the sender and the delivered message.

AbC: Attribute-based Communication

AbC: Attribute-based Communication

Components:

$$C ::= \quad [\Gamma \triangleright P]_\iota \mid C_1 || C_2$$

Γ is a function mapping attributes in $\mathbb{A}$ to values in $\mathbb{VAL}$.

AbC: Attribute-based Communication

Components:

$$C ::= \quad [\Gamma \triangleright P]_\iota \mid C_1 || C_2$$

Processes:

$$\begin{aligned} P ::= \quad & \mathsf{skip} \mid (\tilde{e})@\pi.P \mid \pi(\tilde{x}) \Rightarrow P \mid [\tilde{a} \leftarrow \tilde{e}]P \\ \mid \quad & \langle \pi \rangle P \mid P_1 + P_2 \mid P_1 | P_2 \mid K \\ \pi ::= \quad & tt \mid ff \mid e_1 \bowtie e_2 \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \neg\pi \\ e ::= \quad & v \mid x \mid a \mid \mathsf{this}.a \end{aligned}$$

AbC: Attribute-based Communication

Components:

$$C ::= \quad [\Gamma \triangleright P]_\iota \mid C_1 || C_2$$

Processes:

$$\begin{array}{rl} P ::= & \mathsf{skip} \mid (\tilde{e})@\pi.P \mid \pi(\tilde{x}) \Rightarrow P \mid [\tilde{a} \leftarrow \tilde{e}]P \\ \mid & \langle \pi \rangle P \mid P_1 + P_2 \mid P_1 | P_2 \mid K \\ \pi ::= & tt \mid ff \mid e_1 \bowtie e_2 \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \neg\pi \\ e ::= & v \mid x \mid a \mid \mathsf{this}.a \end{array}$$

Output: the evaluation of $\tilde{e}$ is sent to all components satisfying π.

AbC: Attribute-based Communication

Components:

$$C ::= \quad [\Gamma \triangleright P]_\iota \mid C_1 || C_2$$

Processes:

$$\begin{array}{rl} P ::= & \mathsf{skip} \mid (\tilde{e})@\pi.P \mid \pi(\tilde{x}) \Rightarrow P \mid [\tilde{a} \leftarrow \tilde{e}]P \\ \mid & \langle\pi\rangle P \mid P_1 + P_2 \mid P_1 | P_2 \mid K \\ \pi ::= & tt \mid ff \mid e_1 \bowtie e_2 \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \neg\pi \\ e ::= & v \mid x \mid a \mid \mathsf{this}.a \end{array}$$

Input: a value v is received from a component with attributes Γ if Γ satisfies $\pi[v/x]$.

AbC: Attribute-based Communication

Components:

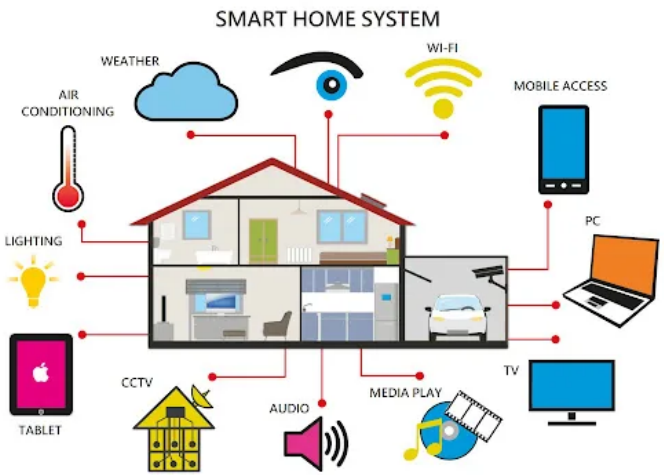
$$C ::= \quad [\Gamma \triangleright P]_\iota \mid C_1 || C_2$$

Processes:

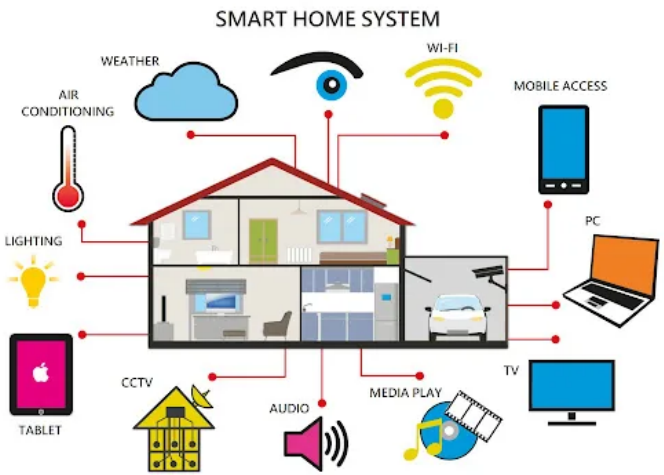
$$\begin{array}{rl} P ::= & \mathsf{skip} \mid (\tilde{e})@\pi.P \mid \pi(\tilde{x}) \Rightarrow P \mid [\tilde{a} \leftarrow \tilde{e}]P \\ \mid & \langle\pi\rangle P \mid P_1 + P_2 \mid P_1 | P_2 \mid K \\ \pi ::= & tt \mid ff \mid e_1 \bowtie e_2 \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \neg\pi \\ e ::= & v \mid x \mid a \mid \mathsf{this}.a \end{array}$$

Attribute update: Attributes $\tilde{a}$ are set to $\tilde{e}$.

An example: Smart Building...

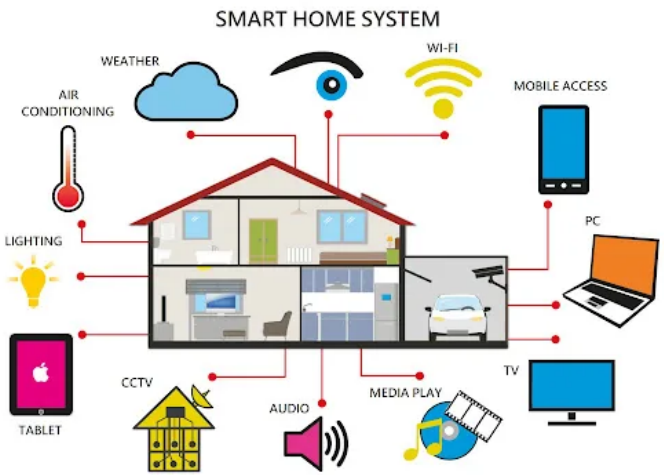


An example: Smart Building...



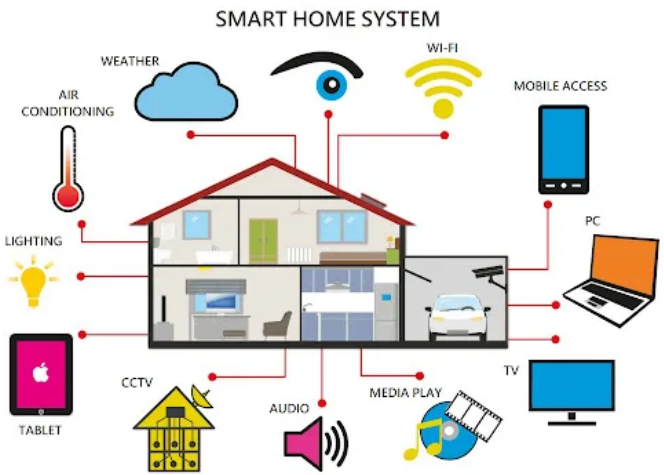
- Sensors send perceived data to controllers.

An example: Smart Building...



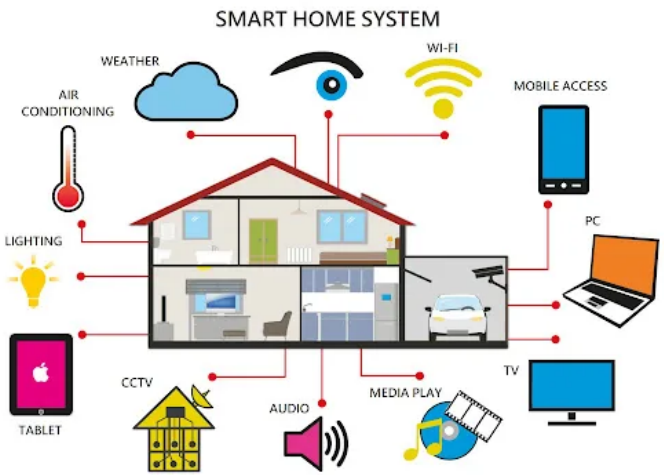
- Sensors send perceived data to controllers.
- Each controller filters received data according to sensor attributes, such as type, location,...

An example: Smart Building...



- Sensors send perceived data to controllers.
- Each controller filters received data according to sensor attributes, such as type, location,...
- A controller can send data to actuators to perform specific tasks.

An example: Smart Building...



- Sensors send perceived data to controllers.
- Each controller filters received data according to sensor attributes, such as type, location,...
- A controller can send data to actuators to perform specific tasks.

Remark: new sensors, actuators and controllers can be transparently integrated!

However...

However...

...a malicious agent can participate in the communication to eavesdrop on potentially confidential data or inject fake data.

However...

...a malicious agent can participate in the communication to eavesdrop on potentially confidential data or inject fake data.

Indeed, in AbC communications are anonymous, and no mechanism is available to restrict/protect data.

However...

...a malicious agent can participate in the communication to eavesdrop on potentially confidential data or inject fake data.

Indeed, in AbC communications are anonymous, and no mechanism is available to restrict/protect data.

Questions:

- Can we integrate security mechanisms in AbC?

However...

...a malicious agent can participate in the communication to eavesdrop on potentially confidential data or inject fake data.

Indeed, in AbC communications are anonymous, and no mechanism is available to restrict/protect data.

Questions:

- Can we integrate security mechanisms in AbC?
- Can we preserve the openendedness of AbC?

However...

...a malicious agent can participate in the communication to eavesdrop on potentially confidential data or inject fake data.

Indeed, in AbC communications are anonymous, and no mechanism is available to restrict/protect data.

Questions:

- Can we integrate security mechanisms in AbC?
- Can we preserve the openendedness of AbC?
- Can we support distributed execution while guarantee preservations of security properties?

Notations. . .

Notations...

Let $\mathbb{K} \subset \mathbb{VAL}$ denote the set of *cryptographic keys*...

- for any $v \in \mathbb{VAL}$ and a key $\kappa \in \mathbb{K}$, $[\!: v :\!]_\kappa = enc(v, \kappa)$ denote *ciphertext* obtained from the encryption of v with the key κ.

Notations...

Let $\mathbb{K} \subset \mathbb{VAL}$ denote the set of *cryptographic keys*...

- for any $v \in \mathbb{VAL}$ and a key $\kappa \in \mathbb{K}$, $[\!: v :\!]_\kappa = enc(v, \kappa)$ denote *ciphertext* obtained from the encryption of v with the key κ.
- given a *ciphertext* $[\!: v :\!]_{\kappa_1}$,

$$dec([\!: v :\!]_{\kappa_1}, \kappa_2) = \begin{cases} v & \kappa_1 \leftrightarrow \kappa_2 \\ \bot_v & \text{otherwise} \end{cases}$$

Notations...

Let $\mathbb{K} \subset \mathbb{VAL}$ denote the set of *cryptographic keys*...

- for any $v \in \mathbb{VAL}$ and a key $\kappa \in \mathbb{K}$, $[\!: v :\!]_\kappa = enc(v, \kappa)$ denote *ciphertext* obtained from the encryption of v with the key κ.
- given a *ciphertext* $[\!: v :\!]_{\kappa_1}$,

$$dec([\!: v :\!]_{\kappa_1}, \kappa_2) = \begin{cases} v & \kappa_1 \leftrightarrow \kappa_2 \\ \bot_v & \text{otherwise} \end{cases}$$

Notations...

Let $\mathbb{K} \subset \mathbb{VAL}$ denote the set of *cryptographic keys*...

- for any $v \in \mathbb{VAL}$ and a key $\kappa \in \mathbb{K}$, $[\!: v :\!]_\kappa = enc(v, \kappa)$ denote *ciphertext* obtained from the encryption of v with the key κ.
- given a *ciphertext* $[\!: v :\!]_{\kappa_1}$,

$$dec([\!: v :\!]_{\kappa_1}, \kappa_2) = \begin{cases} v & \kappa_1 \leftrightarrow \kappa_2 \\ \bot_v & \text{otherwise} \end{cases}$$

where $\kappa_1 \leftrightarrow \kappa_2$ is a key matching predicate

- *symmetric encryption*: $\leftrightarrow$ is the identity predicate.
- *asymmetric encryption*: a *secret key* κ^S matches is public counter part κ^P, and viceversa.

Notations...

Let $\mathbb{K} \subset \mathbb{VAL}$ denote the set of *cryptographic keys*...

- for any $v \in \mathbb{VAL}$ and a key $\kappa \in \mathbb{K}$, $[\!: v :\!]_\kappa = enc(v, \kappa)$ denote *ciphertext* obtained from the encryption of v with the key κ.
- given a *ciphertext* $[\!: v :\!]_{\kappa_1}$,

$$dec([\!: v :\!]_{\kappa_1}, \kappa_2) = \begin{cases} v & \kappa_1 \leftrightarrow \kappa_2 \\ \bot_v & \text{otherwise} \end{cases}$$

  where $\kappa_1 \leftrightarrow \kappa_2$ is a key matching predicate

  - *symmetric encryption*: $\leftrightarrow$ is the identity predicate.
  - *asymmetric encryption*: a *secret key* κ^S matches is public counter part κ^P, and viceversa.

Let $\mathbb{I} \subseteq \mathbb{VAL}$ denote the set of *identities*, $\{\!: v :\!\}_\iota$ denote the value v signed by ι.

CrAbC: Cryptographic AbC

In CrAbC, we consider a larger set of expressions to integrate cryptographic primitives:

e	::=	v	Value
	|	$\mathtt{x}$	Variable
	|	a	Attribute
	|	this.a	
	|	e_1 *bop* e_2	Binary Operator
	|	*uop* e_1	Unary Operator
	|	$\mathsf{enc}(e_1, e_2)$	Encryption
	|	$\mathsf{dec}(e_1, e_2)$	Decryption
	|	$\mathsf{sign}(e)$	Signature
	|	$\mathsf{whois}(e)$	Identity Checking
	|	self	Self reference

CrAbC: Cryptographic AbC

Question: Are cryptographic extensions enough?

CrAbC: Cryptographic AbC

Question: Are cryptographic extensions enough?

The new primitives allow us to protect the data exchanged and limit the injection of fake data.

CrAbC: Cryptographic AbC

Question: Are cryptographic extensions enough?

The new primitives allow us to protect the data exchanged and limit the injection of fake data.

Unfortunately, a malicious agent can acquire info about the state of another component. Indeed, predicates can be used to test the values of attributes!

CrAbC: Access Policies

An access policy Π is a function mapping an identity ι and an attribute a with a set of capabilities in $\{\mathsf{i}, \mathsf{o}\}$.

CrAbC: Access Policies

An access policy Π is a function mapping an identity ι and an attribute a with a set of capabilities in $\{\mathsf{i}, \mathsf{o}\}$.

Access policies allow for regulating the usage of attributes:

- if $\mathsf{i} \in \Pi(\iota, \mathsf{a})$ then component ι can use attrubute a to filter incoming messages.
- if $\mathsf{o} \in \Pi(\iota, \mathsf{a})$ then component ι can use attrubute a to dispatch outgoing messages.

CrAbC: Access Policies

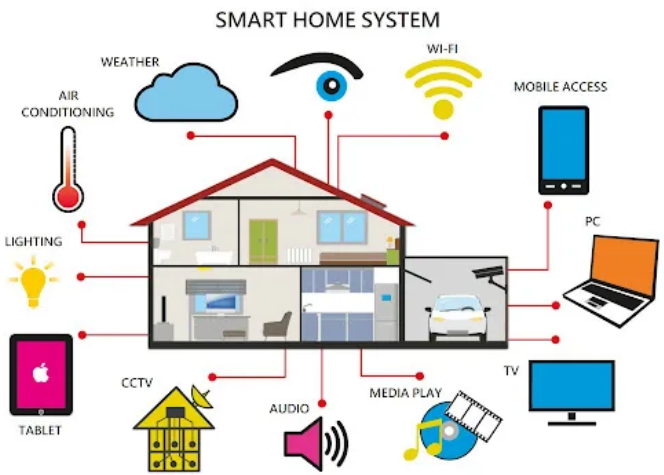
An access policy Π is a function mapping an identity ι and an attribute a with a set of capabilities in $\{\mathsf{i}, \mathsf{o}\}$.

Access policies allow for regulating the usage of attributes:

- if $\mathsf{i} \in \Pi(\iota, \mathsf{a})$ then component ι can use attrubute a to filter incoming messages.
- if $\mathsf{o} \in \Pi(\iota, \mathsf{a})$ then component ι can use attrubute a to dispatch outgoing messages.

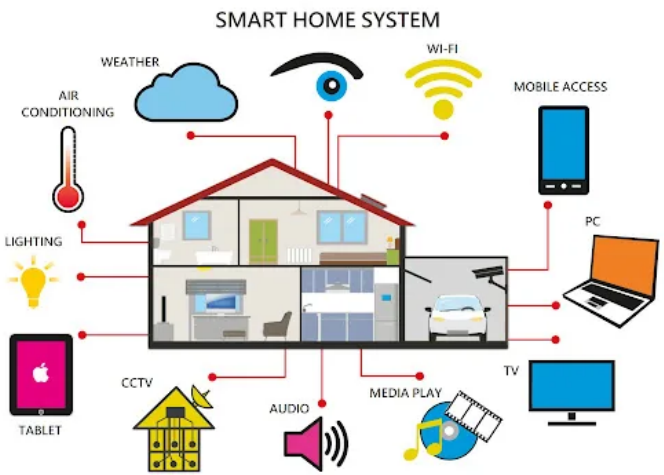
The compliance with a policy Π can be statically checked via a type systems!

An example: Smart Building...



Policies allow us to control the interactions between devices:

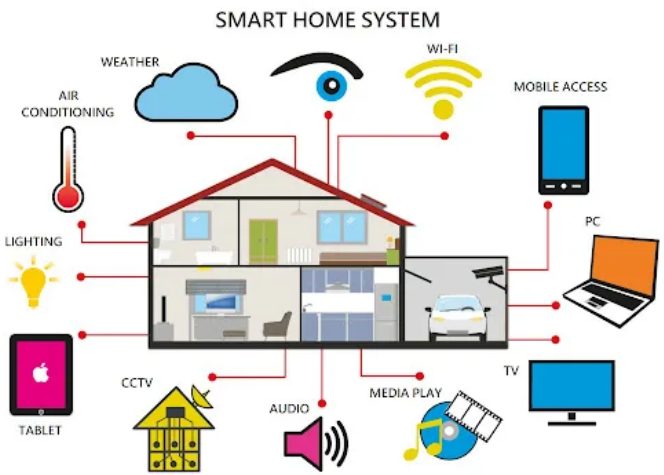
An example: Smart Building...



Policies allow us to control the interactions between devices:

- Only sensors can send data to controllers.

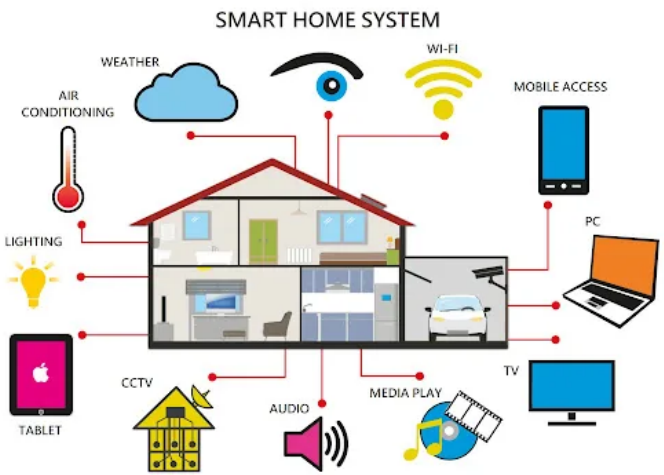
An example: Smart Building...



Policies allow us to control the interactions between devices:

- Only sensors can send data to controllers.
- Only controllers can receive data from sensors.

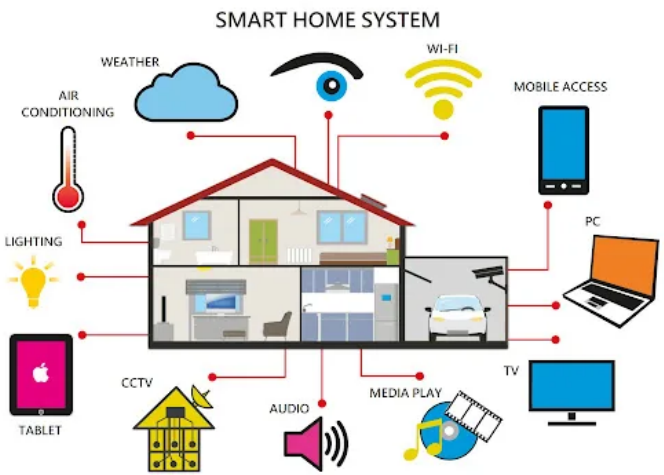
An example: Smart Building...



Policies allow us to control the interactions between devices:

- Only sensors can send data to controllers.
- Only controllers can receive data from sensors.
- Only controllers can send data to actuators.

An example: Smart Building...



Policies allow us to control the interactions between devices:

- Only sensors can send data to controllers.
- Only controllers can receive data from sensors.
- Only controllers can send data to actuators.

New devices can be integrated by registering them in the policies.

Few remarks...

Few remarks...

CRABC allows to program and control distributed opportunistic interactions.

Few remarks...

CRABC allows to program and control distributed opportunistic interactions.

Cryptographic primitives can be used to guarantee data integrity and confidentiality.

Few remarks...

CRABC allows to program and control distributed opportunistic interactions.

Cryptographic primitives can be used to guarantee data integrity and confidentiality.

Policies limit the visibility of attributes and limit information leakage.

Few remarks...

CRABC allows to program and control distributed opportunistic interactions.

Cryptographic primitives can be used to guarantee data integrity and confidentiality.

Policies limit the visibility of attributes and limit information leakage.

Question: How can we implement/execute CRABC?

Secure execution of CrAbC

To support secure execution of CrAbC, we need a framework that...

Secure execution of CrAbC

To support secure execution of CrAbC, we need a framework that...

- Supports secure communication to guarantee that data is not *sniffed*.

Secure execution of CRABC

To support secure execution of CRABC, we need a framework that...

- Supports secure communication to guarantee that data is not *sniffed*.
- Only eligible devices will receive messages (a component cannot cheat on the predicate satisfaction).

Secure execution of CRABC

To support secure execution of CRABC, we need a framework that...

- Supports secure communication to guarantee that data is not *sniffed*.
- Only eligible devices will receive messages (a component cannot cheat on the predicate satisfaction).
- CRABC semantics is preserved.

Secure execution of CRABC

To support secure execution of CRABC, we need a framework that...

- Supports secure communication to guarantee that data is not *sniffed*.
- Only eligible devices will receive messages (a component cannot cheat on the predicate satisfaction).
- CRABC semantics is preserved.

A framework secure attribute-based publish subscribe has been proposed.

Secure Publish-Subscribe

We consider a variant of the standard publish-subscribe communication paradigm in which messages are selected based on attributes.

Secure Publish-Subscribe

We consider a variant of the standard publish-subscribe communication paradigm in which messages are selected based on attributes.

The proposed framework relies on two main entities...

Secure Publish-Subscribe

We consider a variant of the standard publish-subscribe communication paradigm in which messages are selected based on attributes.

The proposed framework relies on two main entities...

- A *Certification Authority*, that has the responsibility of:
  - controlling the identities of communicating agents;
  - regulating the access to attributes;
  - evaluate predicates.

Secure Publish-Subscribe

We consider a variant of the standard publish-subscribe communication paradigm in which messages are selected based on attributes.

The proposed framework relies on two main entities...

- A *Certification Authority*, that has the responsibility of:
  - controlling the identities of communicating agents;
  - regulating the access to attributes;
  - evaluate predicates.
- A *Broker*, that has the responsibility of:
  - coordinating agents' interactions;
  - register subscriptions;
  - store values of agent attributes;
  - deliver messages.

SecPubSub in a nutshel...

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...

  ... data leakage is limited, even if one has access to the broker memory!

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...
  ... data leakage is limited, even if one has access to the broker memory!
- Messages from agents to the Broker are encrypted with the public key of the Certification Authority and signed with the secret key of the agent...

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...
  - ... data leakage is limited, even if one has access to the broker memory!
- Messages from agents to the Broker are encrypted with the public key of the Certification Authority and signed with the secret key of the agent...
  - ... this guarantees data confidentiality and integrity.

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...
  - ... data leakage is limited, even if one has access to the broker memory!
- Messages from agents to the Broker are encrypted with the public key of the Certification Authority and signed with the secret key of the agent...
  - ... this guarantees data confidentiality and integrity.
- Agents update attribute values at the broker that relies on the Certification Authority to obtain the pair (*encrypted attribute*, *encrypted value*) to store.

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...
  - ... data leakage is limited, even if one has access to the broker memory!
- Messages from agents to the Broker are encrypted with the public key of the Certification Authority and signed with the secret key of the agent...
  - ... this guarantees data confidentiality and integrity.
- Agents update attribute values at the broker that relies on the Certification Authority to obtain the pair (*encrypted attribute*, *encrypted value*) to store.
- Similarly, subscriptions are stored encrypted by the Certification Authority at the Broker.

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...
  - ... data leakage is limited, even if one has access to the broker memory!
- Messages from agents to the Broker are encrypted with the public key of the Certification Authority and signed with the secret key of the agent...
  - ... this guarantees data confidentiality and integrity.
- Agents update attribute values at the broker that relies on the Certification Authority to obtain the pair (*encrypted attribute*, *encrypted value*) to store.
- Similarly, subscriptions are stored encrypted by the Certification Authority at the Broker.
- When a message is published, the Broker interacts with the Certification Authority to select the agents where the message must be delivered...

SecPubSub in a nutshel...

- Broker stores attribute values of participating agents. Both attributes and values are encrypted by the Certification Authority...
  - ... data leakage is limited, even if one has access to the broker memory!
- Messages from agents to the Broker are encrypted with the public key of the Certification Authority and signed with the secret key of the agent...
  - ... this guarantees data confidentiality and integrity.
- Agents update attribute values at the broker that relies on the Certification Authority to obtain the pair (*encrypted attribute*, *encrypted value*) to store.
- Similarly, subscriptions are stored encrypted by the Certification Authority at the Broker.
- When a message is published, the Broker interacts with the Certification Authority to select the agents where the message must be delivered...
  - ... policies are checked at this time.

From CrAbC to SecPubSub

The mapping is straightforward:

- outputs are transformed in a publish operation;
- inputs are subscriptions.

From CrAbC to SecPubSub

The mapping is straightforward:

- outputs are transformed in a publish operation;
- inputs are subscriptions.

Correctness of translation has been proved to show that...

- any computation is CrAbC is mimiked by the implementation in SecPubSub;
- computations experienced in SecPubSub are related to ones in CrAbC.

Concluding Remarks

In this talk...

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.
- We have see how to run CrAbC over SecPubSub.

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.
- We have see how to run CrAbC over SecPubSub.

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.
- We have see how to run CrAbC over SecPubSub.

Ongoing work...

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.
- We have see how to run CrAbC over SecPubSub.

Ongoing work...

- A Rocq theory is under development...
  - to prove properties of CrAbC systems;
  - to automatise the proof of correctness of execution of CrAbC over SecPubSub.

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.
- We have see how to run CrAbC over SecPubSub.

Ongoing work...

- A Rocq theory is under development...
  - to prove properties of CrAbC systems;
  - to automatise the proof of correctness of execution of CrAbC over SecPubSub.
- An implementation of SecPubSub in Python is under development.

Concluding Remarks

In this talk...

- We have presented CrAbC, a cryptographic variant of AbC thought to support secure computations.
- We have introduced SecPubSub, a publish-subscribe approach, based on attributes, that guarantee data confidentiality, autenticity and integrity.
- We have see how to run CrAbC over SecPubSub.

Ongoing work...

- A Rocq theory is under development...
  - to prove properties of CrAbC systems;
  - to automatise the proof of correctness of execution of CrAbC over SecPubSub.
- An implementation of SecPubSub in Python is under development.
- We are generalising the approach to consider multiple Certification Authorities and Brokers to avoid single points of failure.


thank
you!