Security and Trust

Gabriele Costa, Fabio Martinelli,
Ilaria Matteucci
(IIT-CNR)
Valérie Issarny and **Rachid Saadi**
(INRIA)

Talk Outline

- Trust model interoperability (Rachid Saadi INRIA-Rocquencourt)
- Security-by-Contract-with-Trust (SxCxT) (Ilaria Matteucci IIT-CNR Pisa)

Trust Outline

- **Introduction**
- Use case
- Trust meta-model
- Trust interoperability

Introduction

- Trust management is becoming a central element in today's open distributed digital environment
- However, existing trust management systems are application and domain dependent
- Hence they often implement different trust models
- As a result, it is really challenging to establish trust relationships across heterogeneous trust management systems

Methodology

- A unified description for trust management systems:
  - **A trust meta-model**
- Automated mediation process that enables interoperability between trust models:
  - **Trust model composition**

Outline

- Introduction
- **Use case**
- Trust meta-model
- Trust interoperability

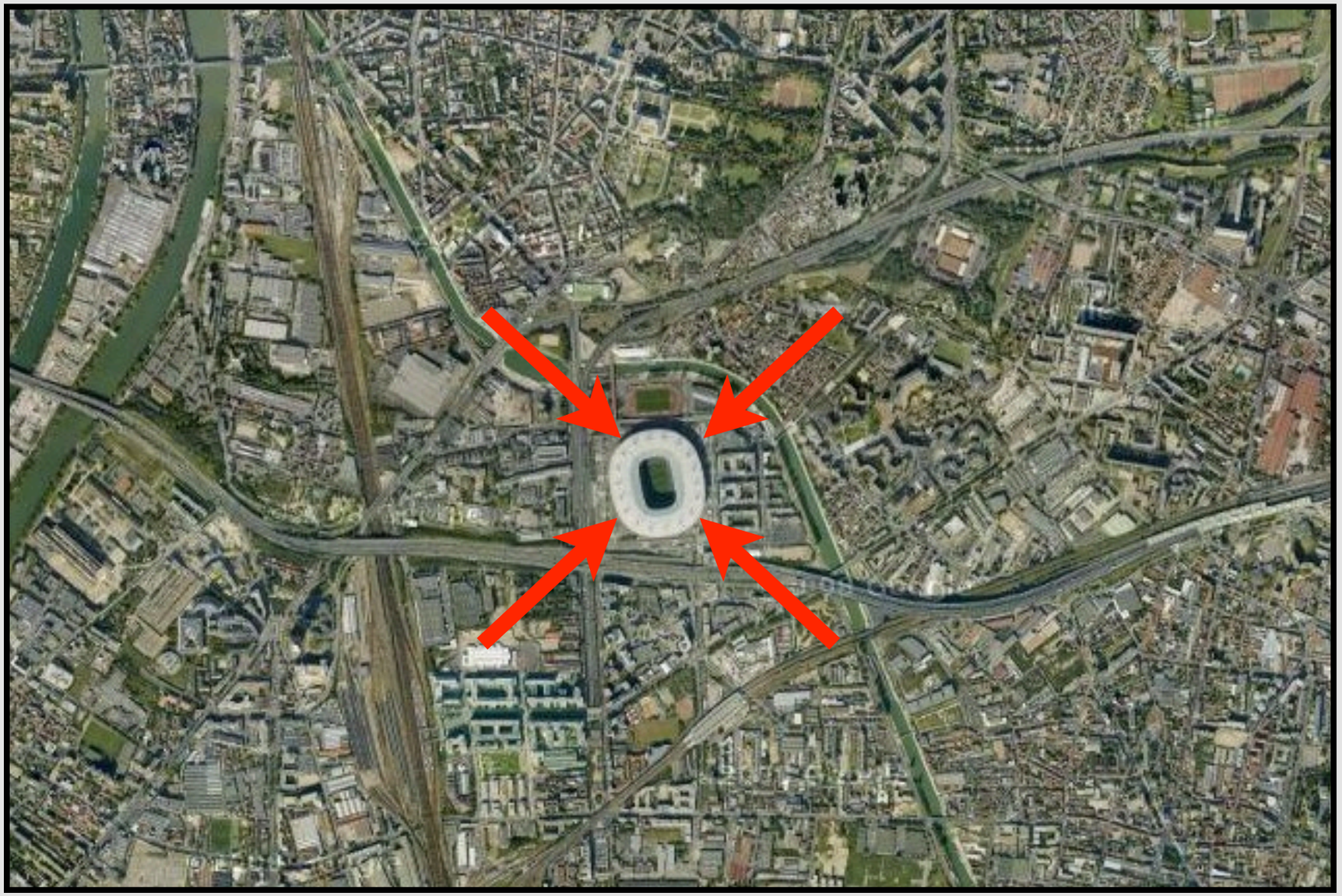
Use Case

Use Case

Use Case

Use Case

Use Case



Use Case

Stadium Security Staff

Use Case

Government Trust Model

Stadium Trust Model

Government Agent

Stadium Guard

Outline

- Introduction
- Use case
- **Trust meta-model**
- Trust interoperability

Trust Meta Model

- As defined in [1]: ***A trustor trusts a trustee with regard to its ability to perform a specific action or to provide a specific service***

- Hence, any trust model may basically be defined in terms of the four following elements:
  - **Trust roles** abstract the representative behaviors of stakeholders
  - **Trust metrics** define how trust is measured
  - **Trust relations** specify trust relationships among stakeholders
  - **Trust operations** define how to compute and assess a relationship

Trust Meta Model

- We formally define the trust meta-model as:
  - $TM =< \mathbb{R}, \mathbb{M}, \mathbb{L}, \mathbb{O} >$ where:
    - $\mathbb{R}$: set of roles
    - $\mathbb{M}$: set of metrics
    - $\mathbb{L}$: set of relations
    - $\mathbb{O}$: set of operations

- We note:
  - $\Diamond\mathbb{S}$ a value from the set $\mathbb{S}$
    - e.g., $s_1 \in \mathbb{S}$
  - $\vee\mathbb{S}$ a disjunction of values from the set $\mathbb{S}$
    - e.g., $s_1 \vee s_2 \vee s_3 / s_1, s_2, s_3 \in \mathbb{S}$
  - $\wedge\mathbb{S}$ a conjunction of values from the set $\mathbb{S}$
    - e.g., $s_1 \wedge s_2 \wedge s_3 / s_1, s_2, s_3 \in \mathbb{S}$

Role Set ($\mathbb{R}$)

- Formally, we define a role as: ***"r =< name::Id >"***

- Stadium Trust model
  - **The Guard**: secures the stadium: r_G=<*name=Guard*>
  - **The Manager**: manages the reputation of the Guards: r_M=<*name=Manager*>
  - **The Chief**: controls and evaluates the Guards: r_C=<*name=Chief*>

- Government Trust model
  - **The Agent**: enforces the government law: r_A=<*name=Agent*>
  - **The Authority:** agent's headquarter (e.g., DGSE, MI6, CIA, etc.): r_T=<*name=Authority*>

Metric Set($\mathbb{M}$)

- Different metrics have been defined to measure trust
- There is no widely recognized way to assign trust values
  - Binary values
  - Semantic label (e.g., high trust, low trust etc.)
  - Numerical range (e.g., [-1,1], [0,n], [0,1], etc.)
  - In many dimensions (e.g., Belief, Disbelief, Uncertainty, etc.)
- Formally, we define a metric as:

m=<name::Id, type::Id>

Example of Trust Metric

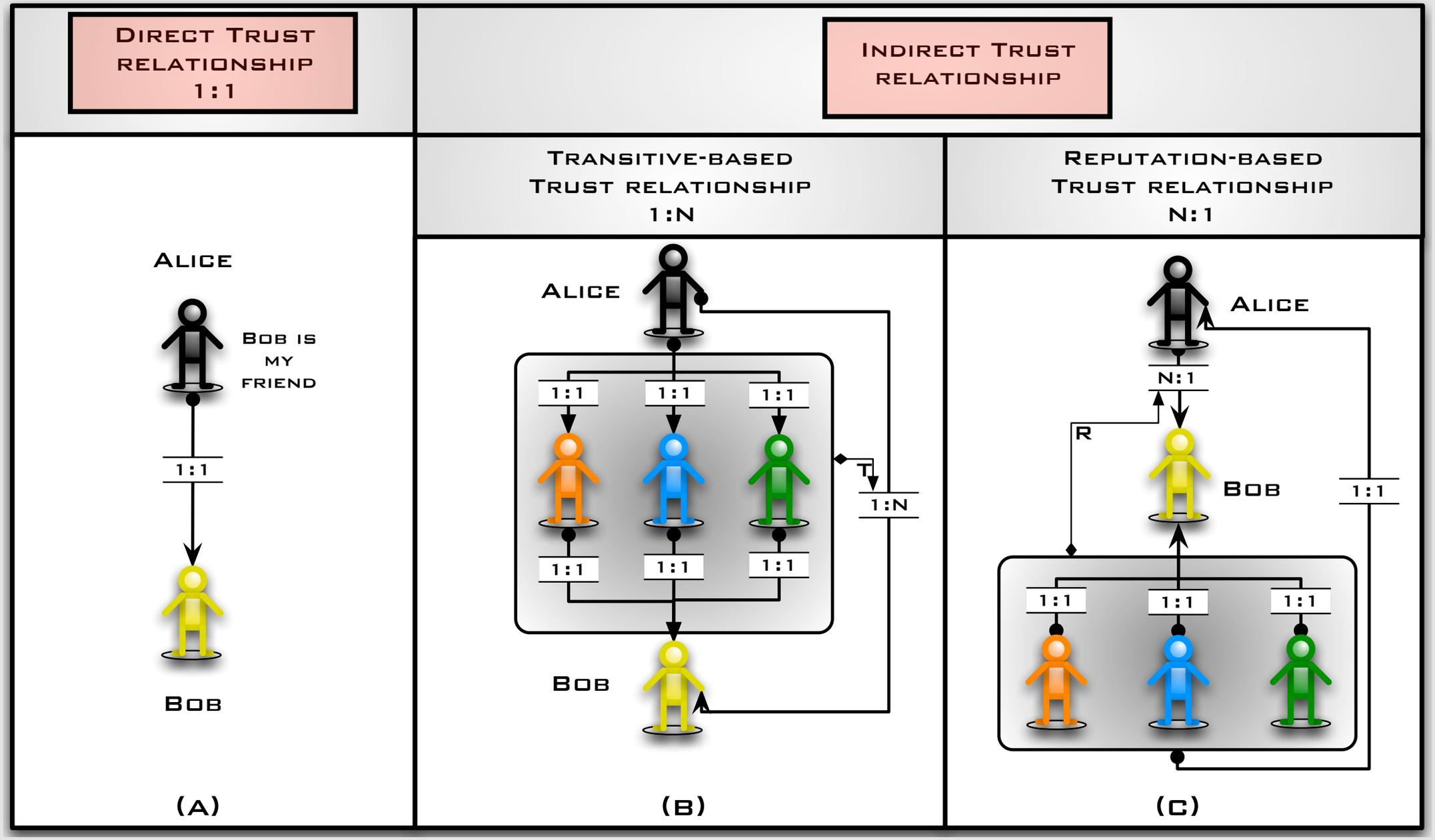
- Stadium Trust model:
  - Guard reputation value: probabilistic value [0,1]
    - i.e., m_{rep}=<name="Reputation", type="Probability">
  - Chief rate: semantic label (Bonus, Penalty)
    - i.e., m_{rat}=<name="Rate", type= "Labels">

- Government Trust model
  - Accreditation levels: an integer interval of values [0..3]
    - i.e., m_{acc}=<name="Accreditation", type="Levels">

Relation Set ($\mathbb{L}$)

In order to formalize trust relationships we first have to identify the different **types** of **relationships**

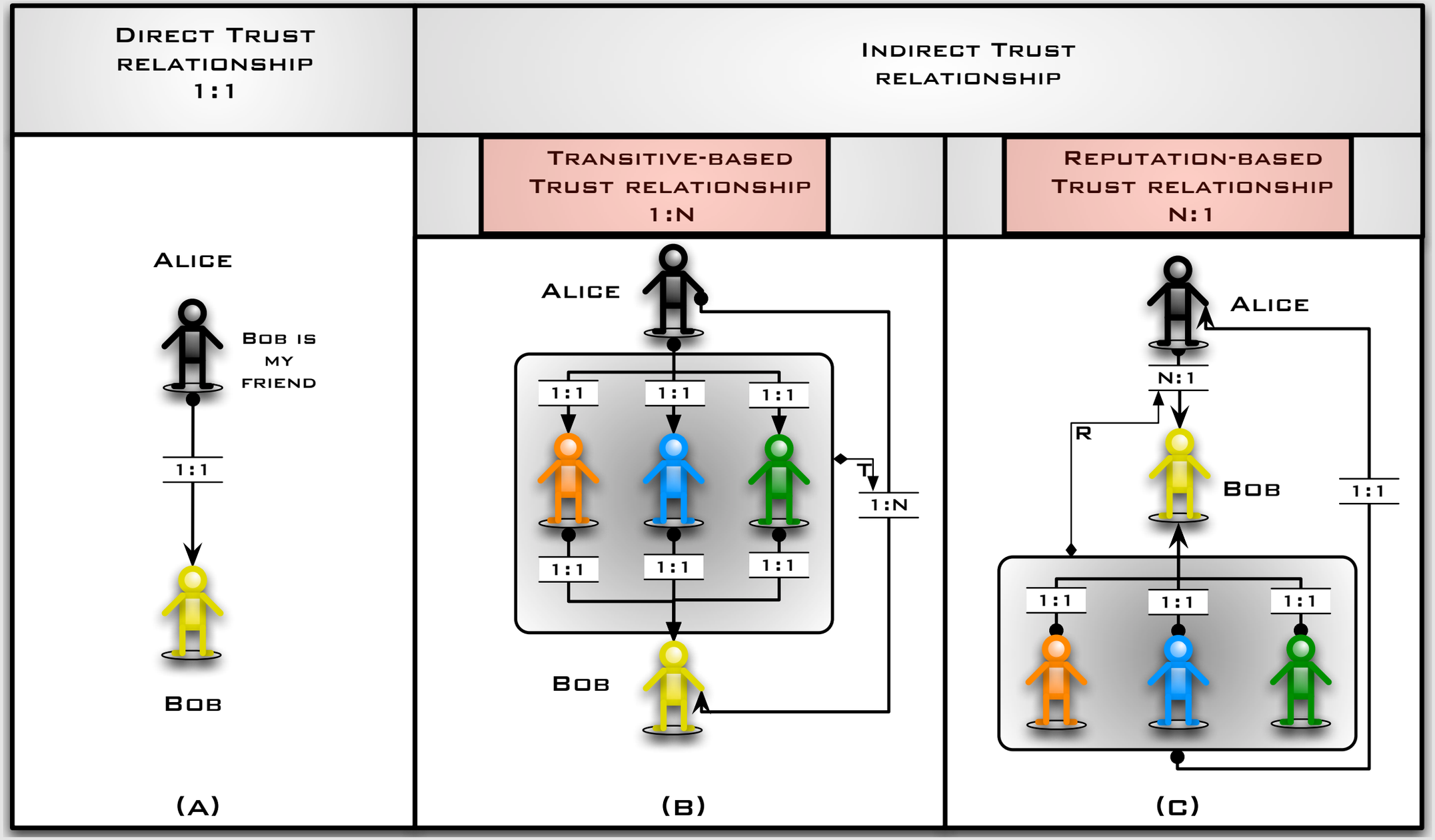
Trust Relation Types



CONNECT

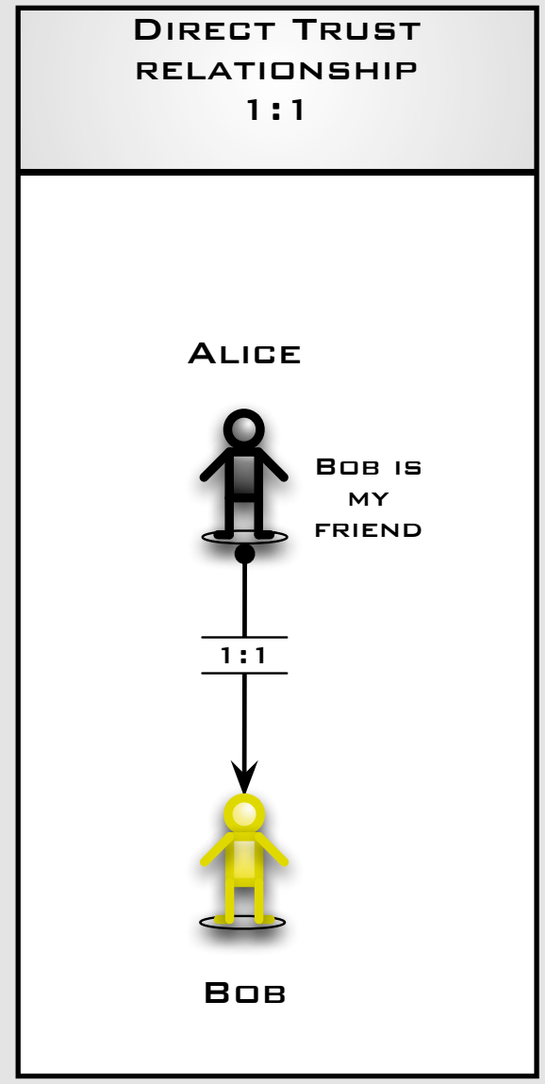
http://connect-forever.eu/

Trust Relation Types



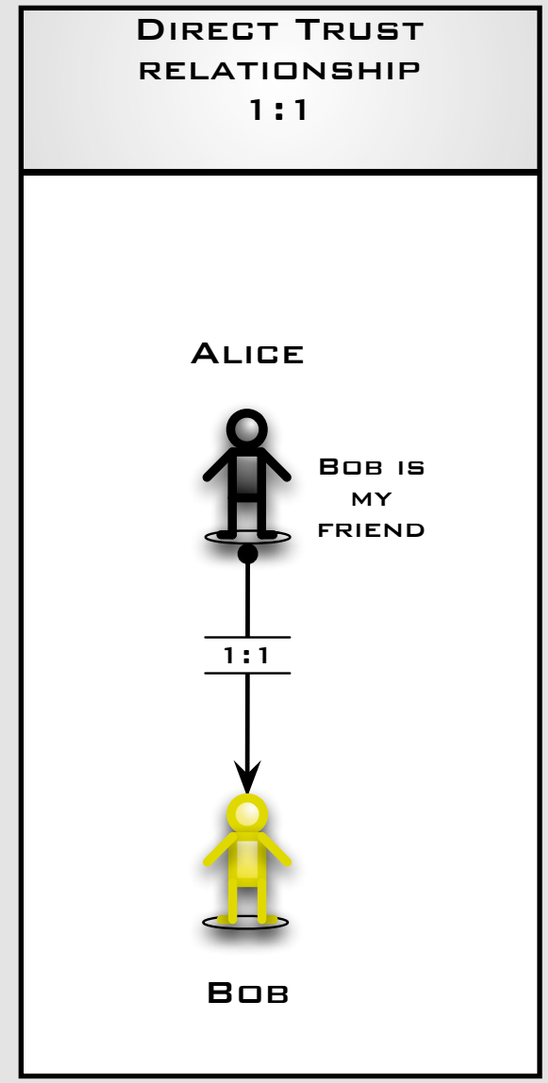
Direct Trust Relation (1:1 one-to-one)

- It is a one-to-one trust relation (**denoted 1:1**) since it defines a direct link from:
  - **(1) trustor to (1) trustee**
- one-to-one trust relations are maintained locally by trustors
- It represents the trustors' opinion regarding their trustees:
  - Objective or Subjective



Direct Trust Relation (1:1 one-to-one)

- Belonging-driven relationship:
  - an employee trusts his company
- Social-driven relationship:
  - trust among friends
- Profit-driven relationship:
  - a person trusts a trader for managing his portfolio
- Quality-driven relationship:
  - a person with 15 years of experience



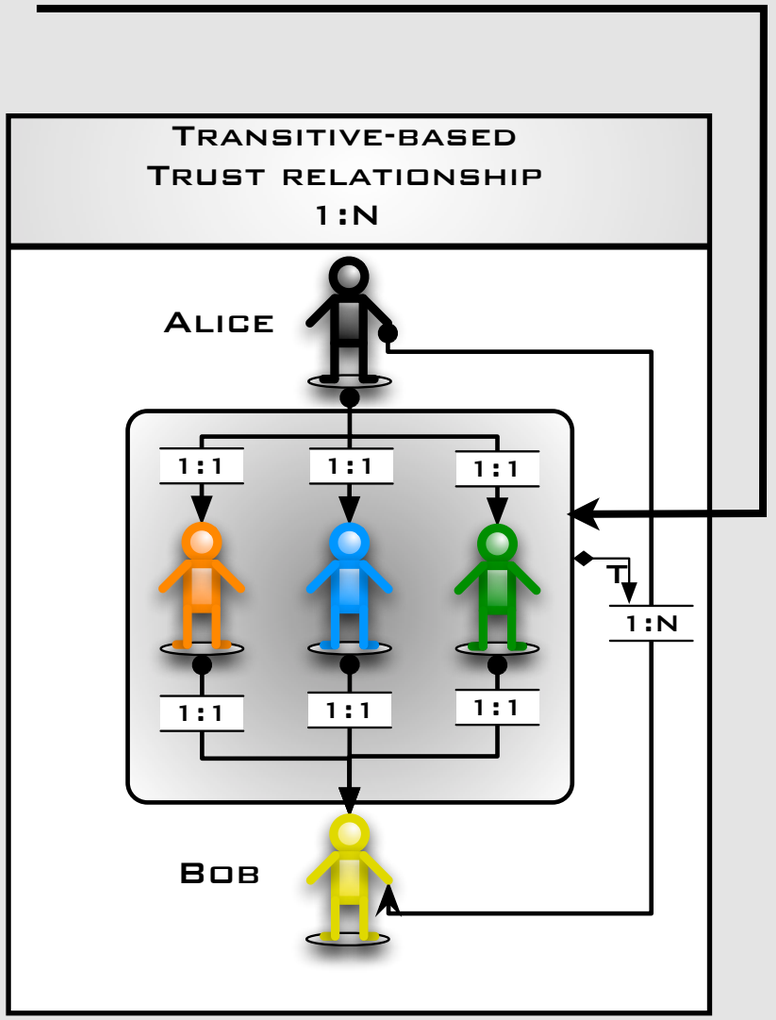
Indirect Trust Relationship

- As opposed to a direct trust relationship, indirect trust relationship enables assessing:
  - **a priori unknown trustee**
  - with **third party's recommendation(s)**

- This can be either:
  - transitive-based
  - reputation-based

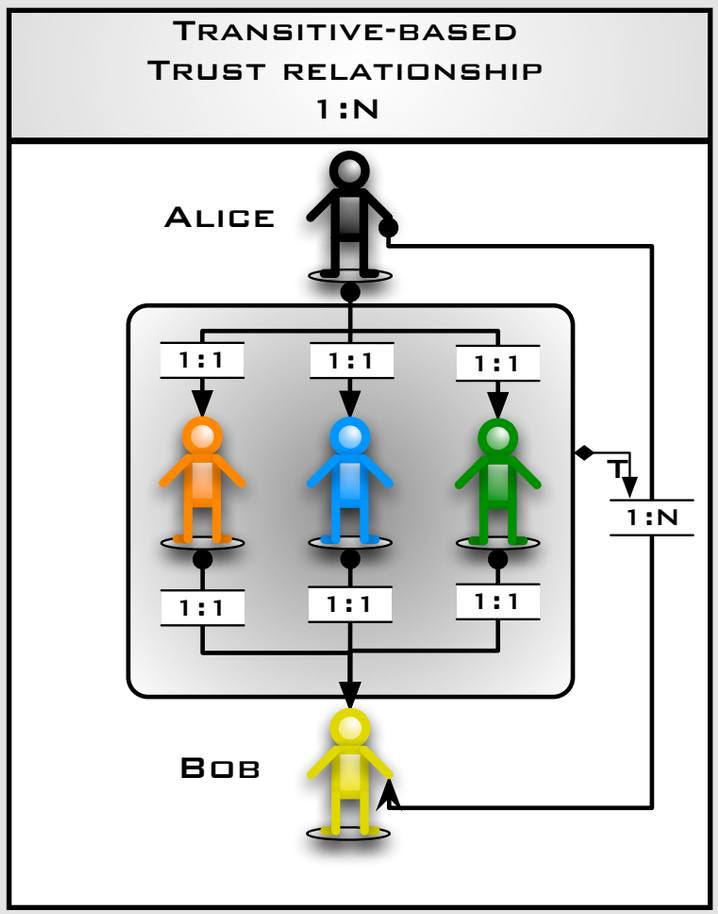
Transitive-Based Trust Relationship (1:N one-to-many)

- Trustor (1) trusts an unknown trustee through a group of trusted **recommenders** ??



Transitive-Based Trust Relationship (1:N one-to-many)

- Trustor (1) trusts an unknown trustee through a group of trusted **recommenders** ??



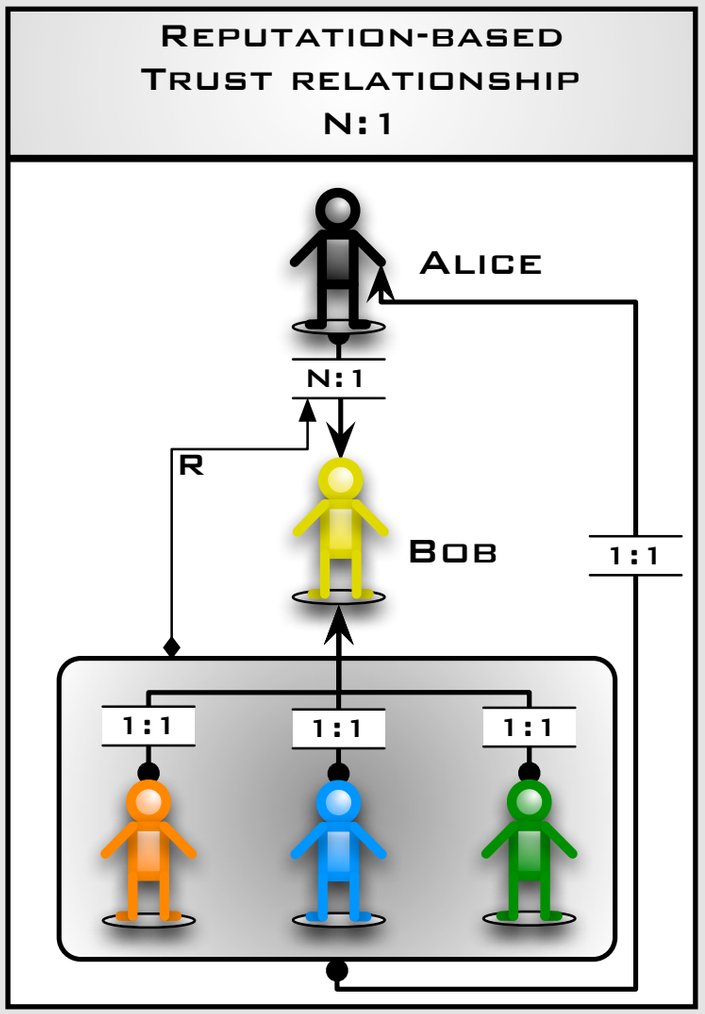
- ***Trustees***
  - The friend of my friend is my friend
- ***Similar trustees***
  - The friend of my similar friend is my friend
- ***Enemies***
  - The enemy of my enemy is my friend

Reputation-Based Trust Relationship (N:1 many-to-one)

- **Recommenders (N)** contribute with their feedbacks (sent to the trustor) to build the reputation of an **unknown trustee (1)**

- We **can trust** unknown **trustees** if they have a **good reputation**.

- We **can distrust** unknown **trustees** if they have a **bad reputation**.

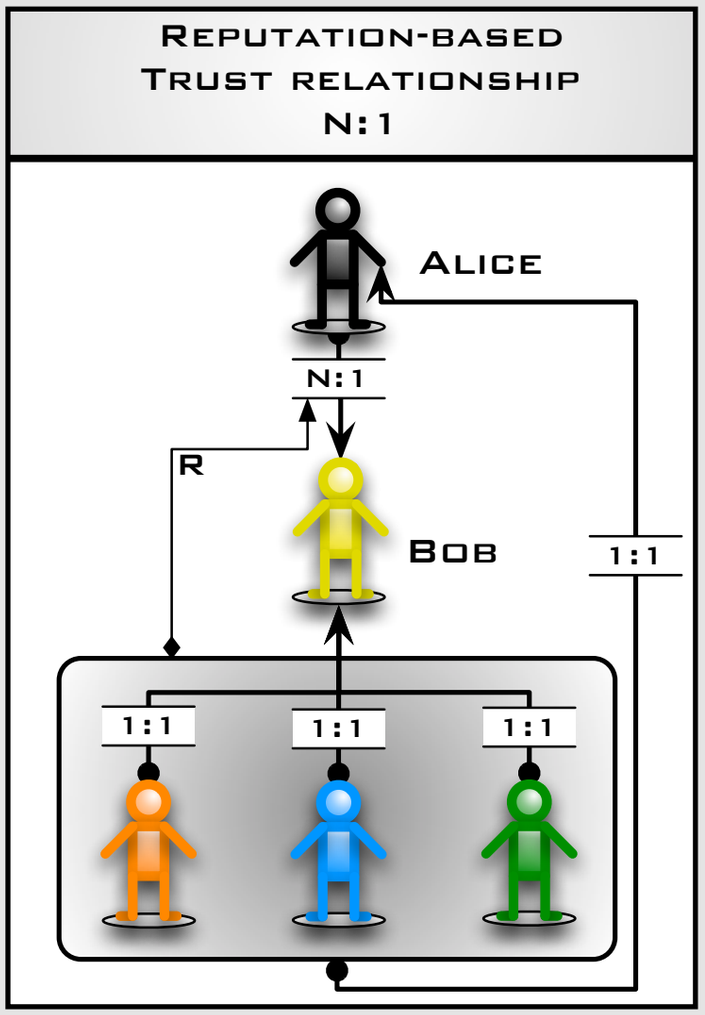


Reputation-Based Trust Relationship (N:1 many-to-one)

- **Recommenders (N)** contribute with their feedbacks (sent to the **trustor**) to build the reputation of an **unknown trustee (1)**

  - ***Centralized*** (ebay like):
    - trusted central server

  - ***Distributed***:
    - randomly define the manager through a hash function, e.g., CAN and Chord
    - replicate the reputation by defining more than one manager



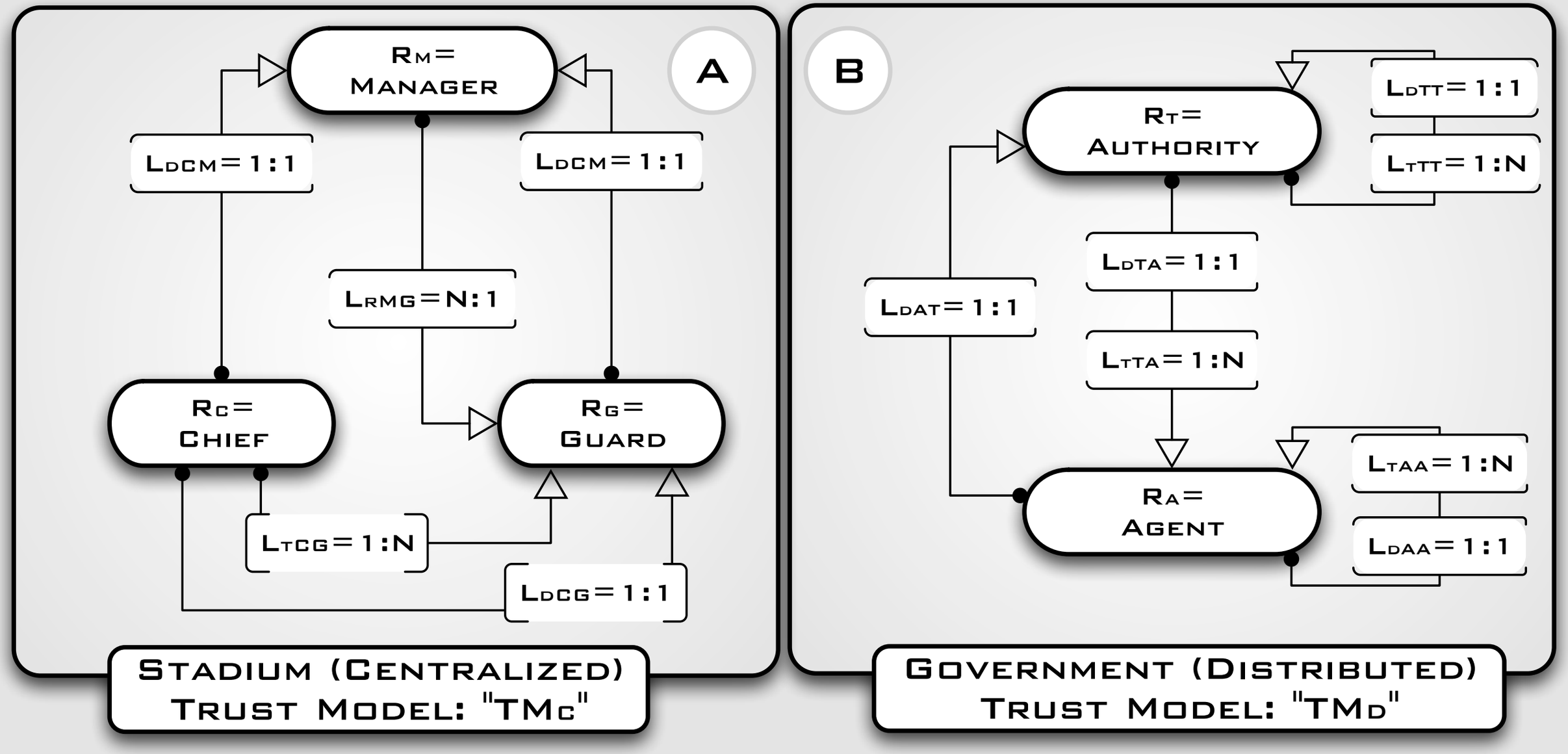
Relation Set ($\mathbb{L}$)

- Recall: ***A trustor trusts a trustee with regard to its ability to perform a specific action or to provide a specific service***
- Thus:
  - A relation set defines all the relations "l" as:

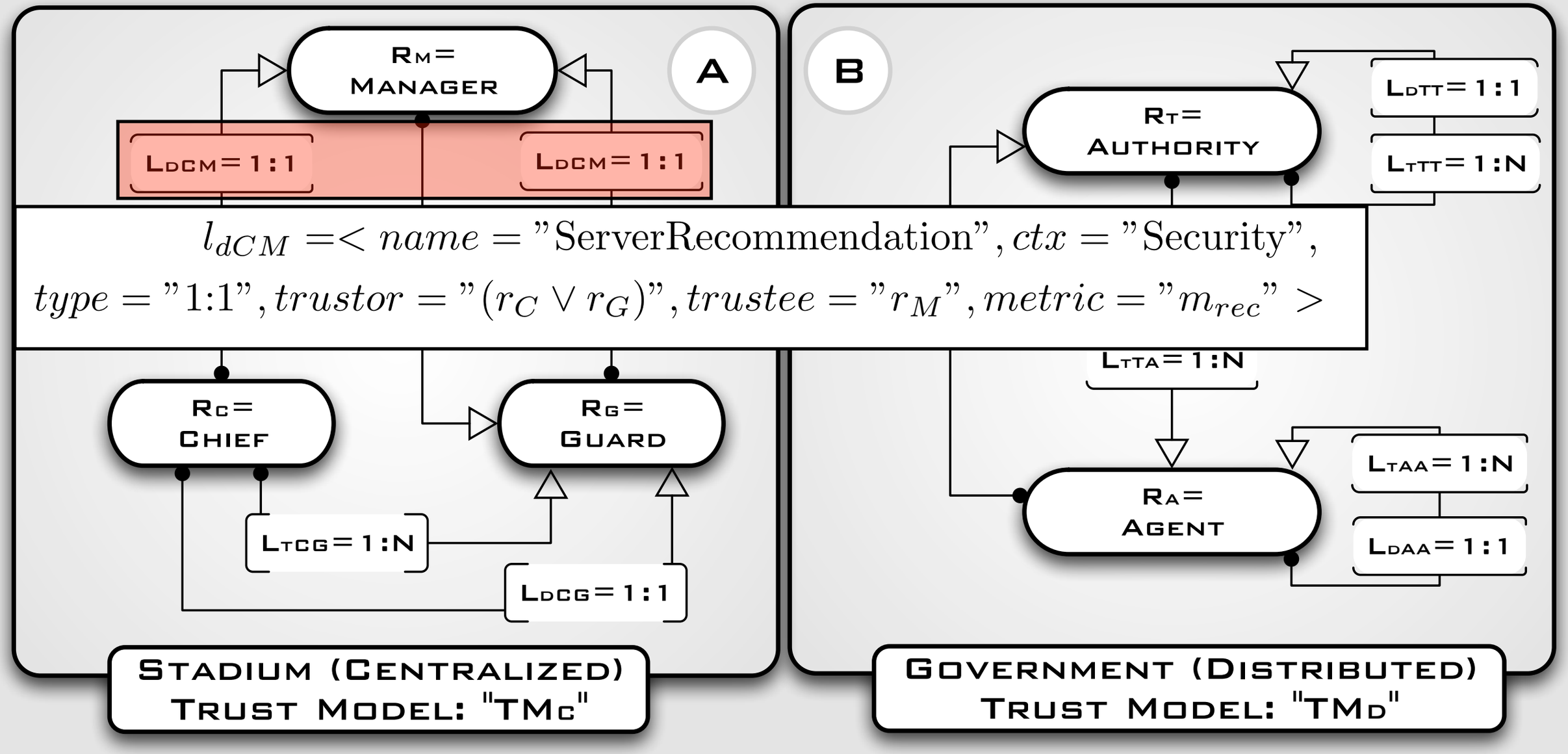
$$l =< \text{name}::Id, \text{ctx}::Id, \text{type}::\Diamond Tl, \text{trustor}::\vee\mathbb{R}, \text{trustee}::\vee\mathbb{R}, \text{metric}::\Diamond\mathbb{M} >$$

  - A relation has a ***name*** and is defined within a specific ***context***
  - A relation can be implemented by different ***trustors*** and ***trustees*** (role)
  - A relation is evaluated by a specific ***metric***
  - A relation has a **type** ***Tl={1:1, 1:N, N:1}***

Example of Trust Relation



Example of Trust Relation



$$l_{dCM} =< name = \text{"ServerRecommendation"}, ctx = \text{"Security"}, type = \text{"1:1"}, trustor = \text{"}(r_C \vee r_G)\text{"}, trustee = \text{"}r_M\text{"}, metric = \text{"}m_{rec}\text{"} >$$

Operation Set ($\mathbb{O}$)

- The operation set includes the operations that can be performed by an entity over relations to assess the trustworthiness of another entity

- We formally define an operation:

$$o =< \text{name}::Id, \text{host}::\vee\mathbb{R}, \text{type}::\Diamond To, \text{input}::\wedge\mathbb{L}, \text{output}::\Diamond\mathbb{L}, \text{call}::\wedge\mathbb{O} >$$

- Where
  - ***name*** identifies an operation
  - ***host*** specifies the role(s) that hosts (executes) the operation
  - ***input*** gives the trust relations that are required to perform an assessment operation and the output gives the resulting trust relation
  - ***call*** denotes a continuation
  - ***type*** defines the type of operation

Operation Set ($\mathbb{O}$)

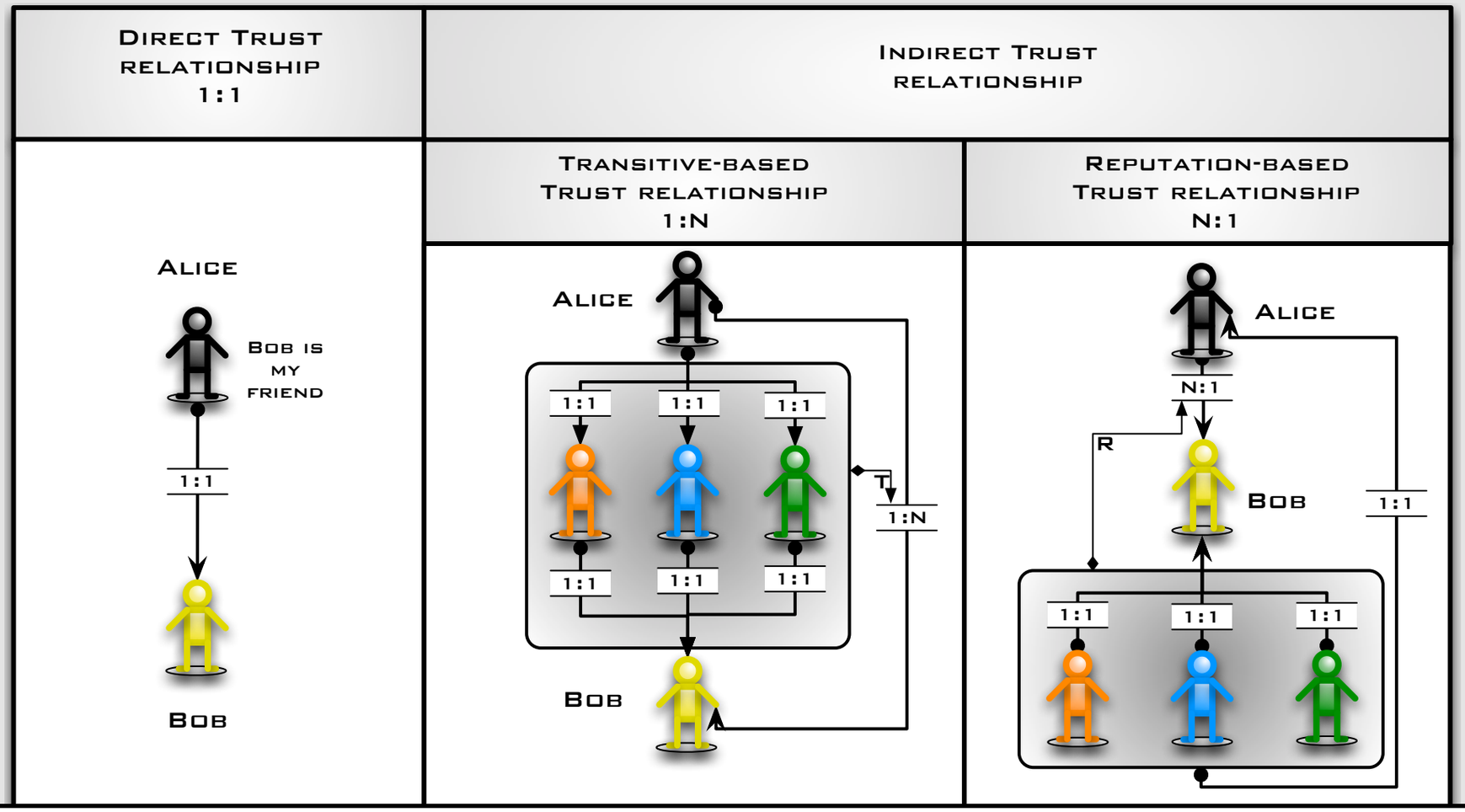
- The operation set includes the operations that can be performed by an entity over relations to assess the trustworthiness of another entity

- We formally define an operation:

$$o =< \text{name}::Id, \text{host}::\vee\mathbb{R}, \text{type}::\Diamond To, \text{input}::\wedge\mathbb{L}, \text{output}::\Diamond\mathbb{L}, \text{call}::\wedge\mathbb{O} >$$

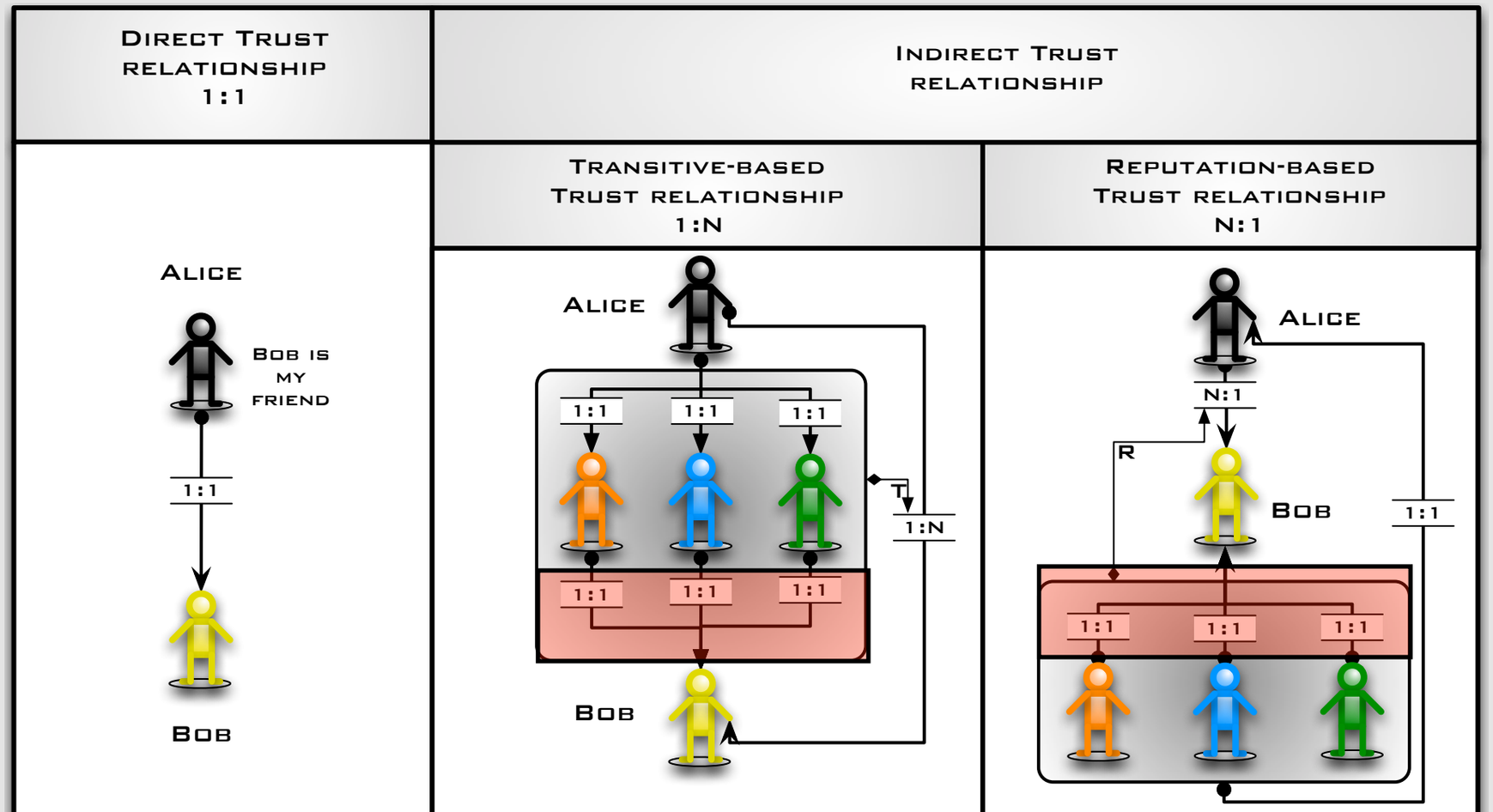
- Where
  - ***name*** identifies an operation
  - ***host*** specifies the role(s) that hosts (executes) the operation
  - ***input*** gives the trust relations that are required to perform an assessment operation and the output gives the resulting trust relation
  - ***call*** denotes a continuation
  - ***type* defines the type of operation ??**

Trust Operation Types



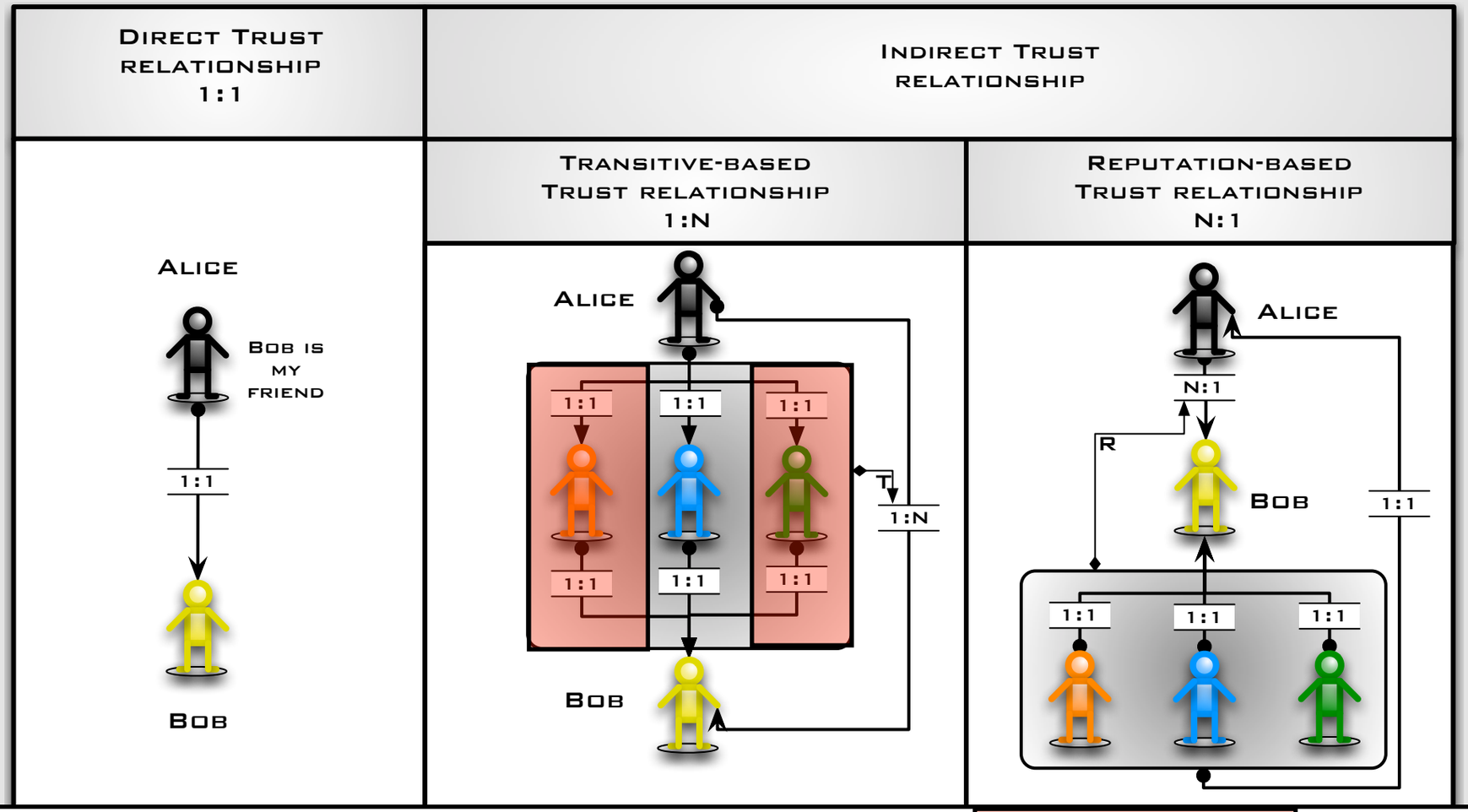
	Bootstrapping	Aggregation	Concatenation	Refreshing
One-to-One (1:1)	X			X
One-to-Many (1:N)		X	X	
Many-to-One (N:1)	X	X		X

Trust Operation Types



	Bootstrapping	Aggregation	Concatenation	Refreshing
One-to-One (1:1)	X			X
One-to-Many (1:N)		X	X	
Many-to-One (N:1)	X	X		X

Trust Operation Types



	Bootstrapping	Aggregation	Concatenation	Refreshing
One-to-One (1:1)	X			X
One-to-Many (1:N)		X	X	
Many-to-One (N:1)	X	X		X

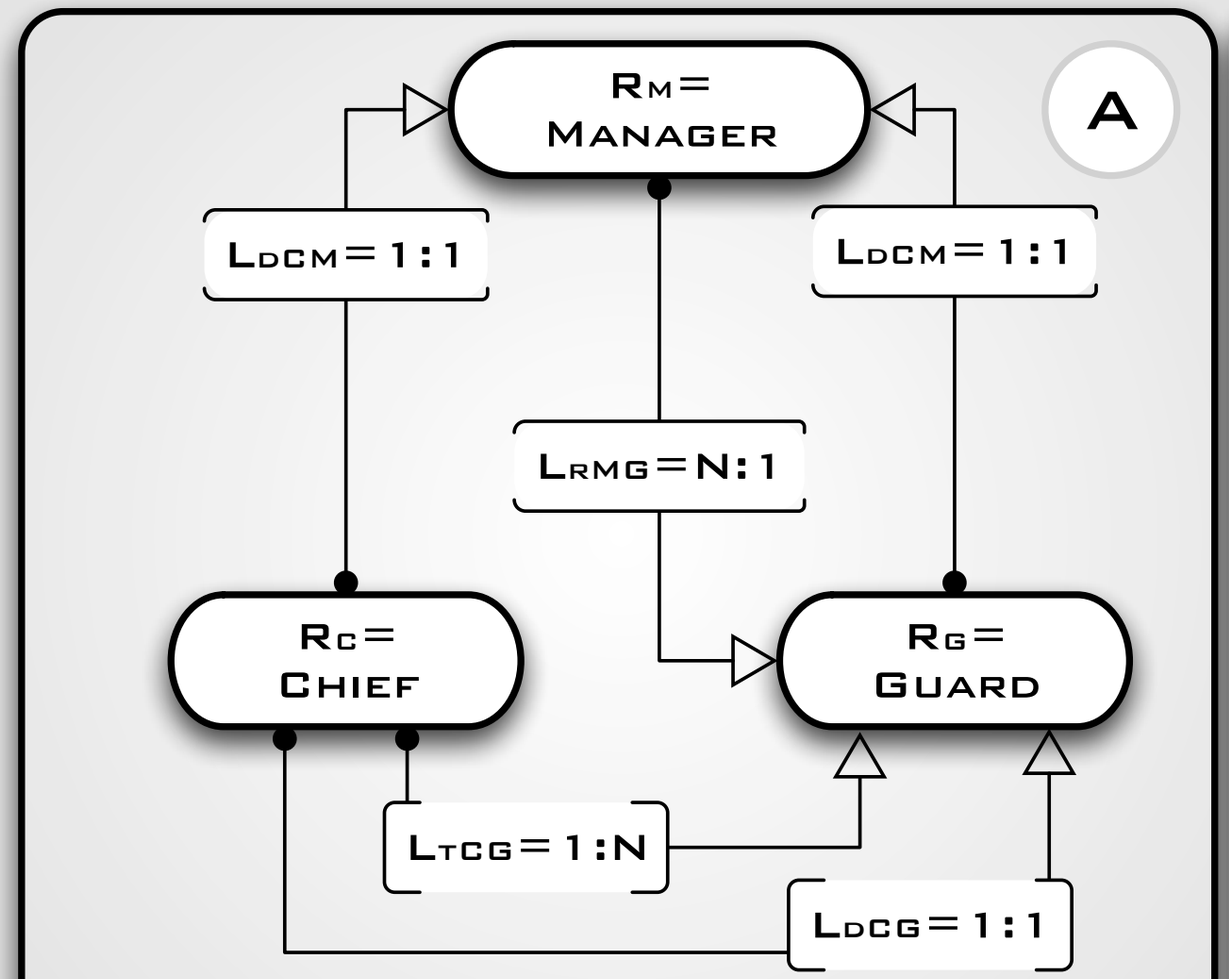
Bootstrapping

- Initially, a trust model evaluates trust relationships with a fixed value (often low)
- Infer unknown trust values from known ones of similar or correlative contexts
  - If no prior related context exists, these solutions can not infer trust
- Infer trust by monitoring social activity (proximity)

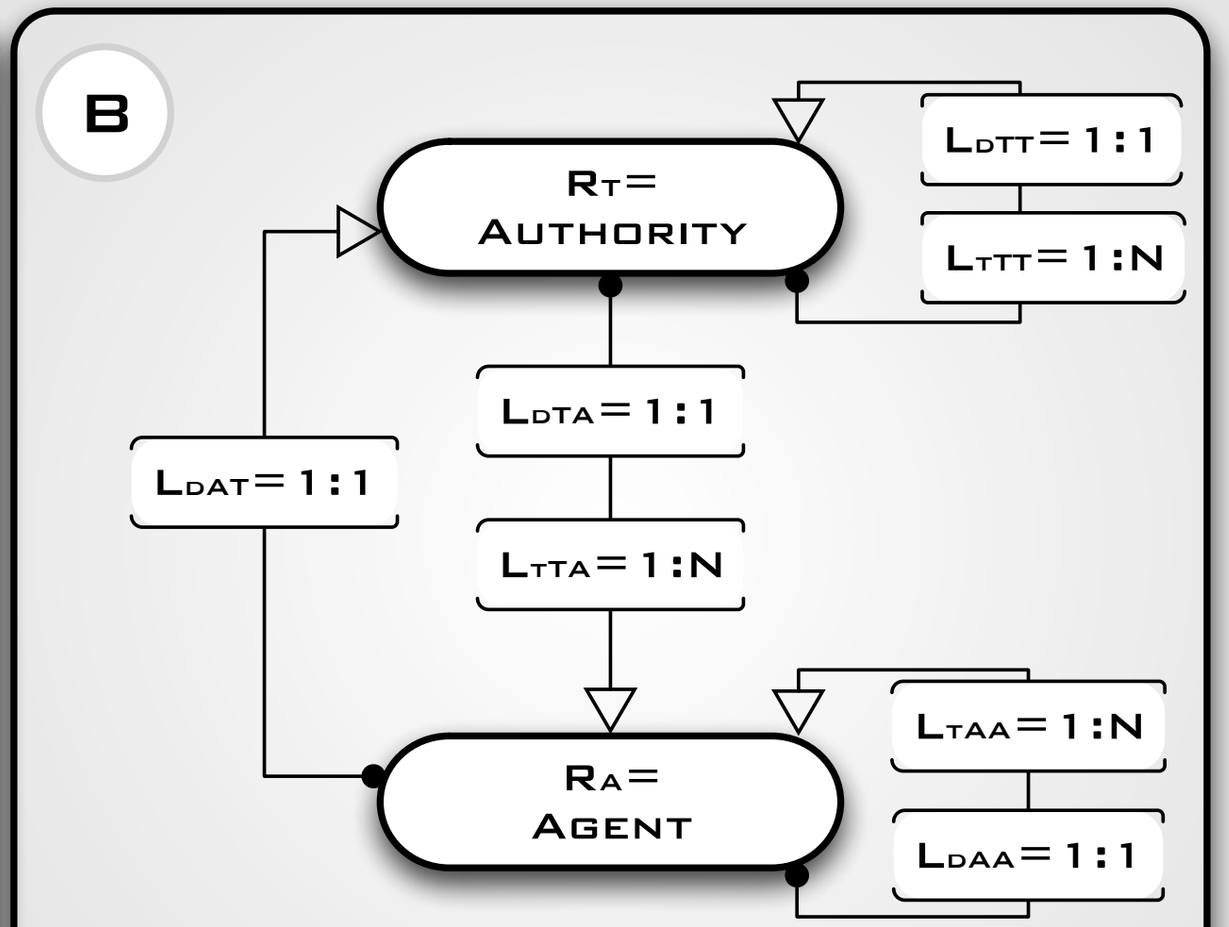
Concatenation & Aggregation

- The multiplication (value[0,1])
- The average
- The Belief model (Belief, Disbelief, Uncertainty)
- The Bayesian systems (binary rating: positive, negative)
- The fuzzy logic models (linguistically fuzzy concepts)
- The experience models (weight the present actions with respect to the past history): $Trust = \alpha * old + (1 - \alpha) * new$

Example of Operations

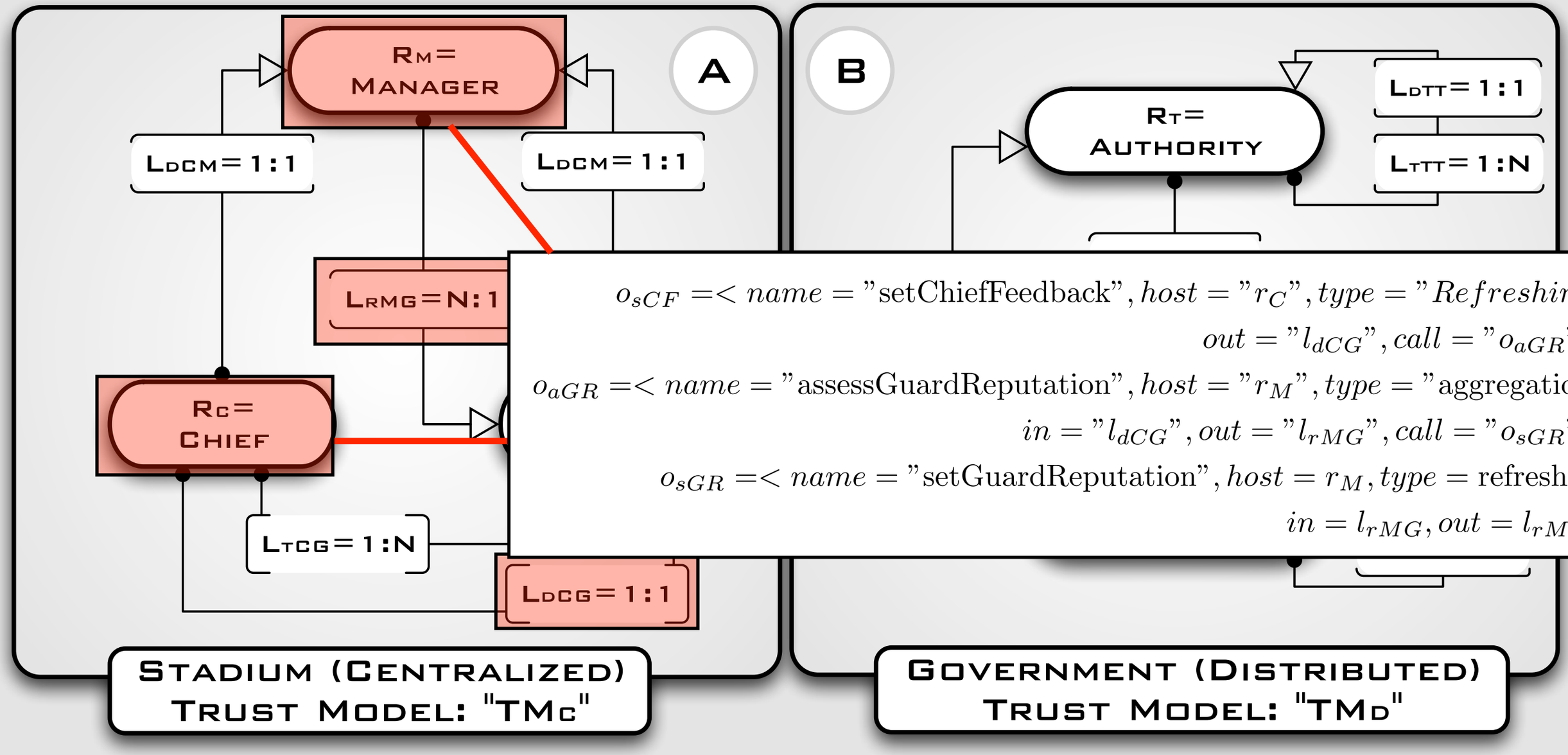


STADIUM (CENTRALIZED) TRUST MODEL: "TMC"



GOVERNMENT (DISTRIBUTED) TRUST MODEL: "TMD"

Example of Operations



Outline

- Introduction
- Use case
- Trust meta-model
- **Trust interoperability**

Trust Models Interoperability

- Objectives:
  - ***Interoperability***
    - Provide **roles** of a given model **(target model TM_T)** the ability to **assess roles** that **pertain** to another trust model **(source model TM_S)**
      - For instance:
        - Guards should help Agents to find suspects.
        - This requires that any Agent be able to assess the accreditation of Guards.
  - ***Transparency***

Trust Models Interoperability

- Objectives:
  - ***Interoperability*** **Mapping rules**
    - Provide **roles** of a given model **(target model TM_T)** the ability to **assess roles** that **pertain** to another trust model **(source model TM_S)**
      - For instance:
        - Guards should help Agents to find suspects.
        - This requires that any Agent be able to assess the accreditation of Guards.
  - ***Transparency*** **Mediation**

Trust Models Interoperability

- Objectives:
  - ***Interoperability*** **Mapping rules**
    - Provide **roles** of a given model **(ta… …odel TM_T)** the ability to **assess roles** that **per…** another trust model **(source model TM_S)**
      - For instance:
        - Guards should h… …ts to find suspects.
        - This requires… …y Agent be able to assess the accreditation of Guards.
  - ***Tra… …ency*** **Mediation**

Trust model composition

Mapping Rule Set (Ψ)

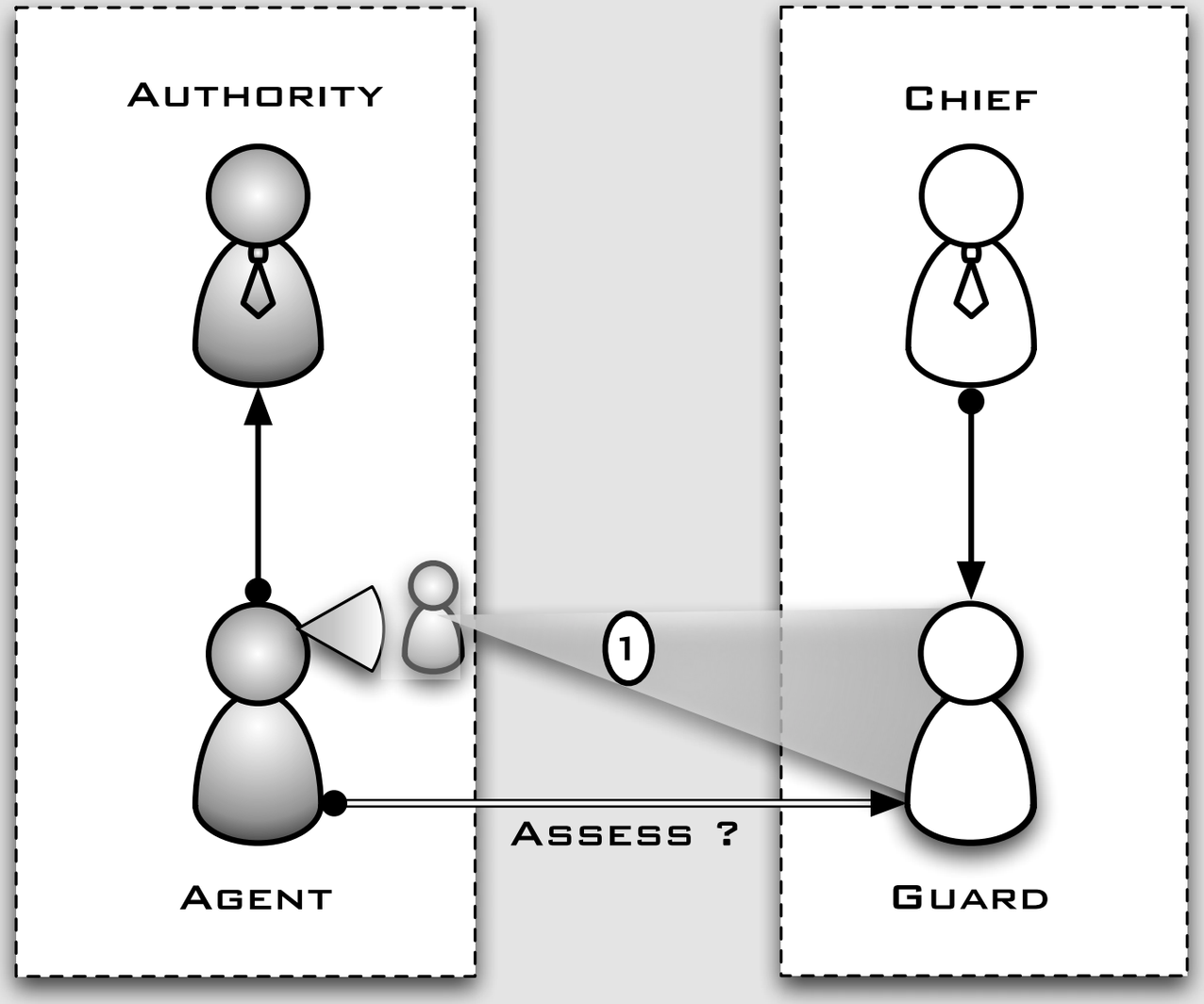
- Given two trust models: *TM_x* and *TM_y*

$$\Psi_{xy} = \{\psi^k_{ST} / \psi^k_{ST} = (r_s : TM_S) \triangleright (r_t : TM_T)\}$$
$$Where\ S, T \in \{x, y\},\ S \neq T$$

  - Each mapping rule ψ^k_{ST} associates :
    - a source role (r_s) with a target role (r_t)
    - to define that the source role (r_s) of (TM_S) plays the target role (r_t) in (TM_T).

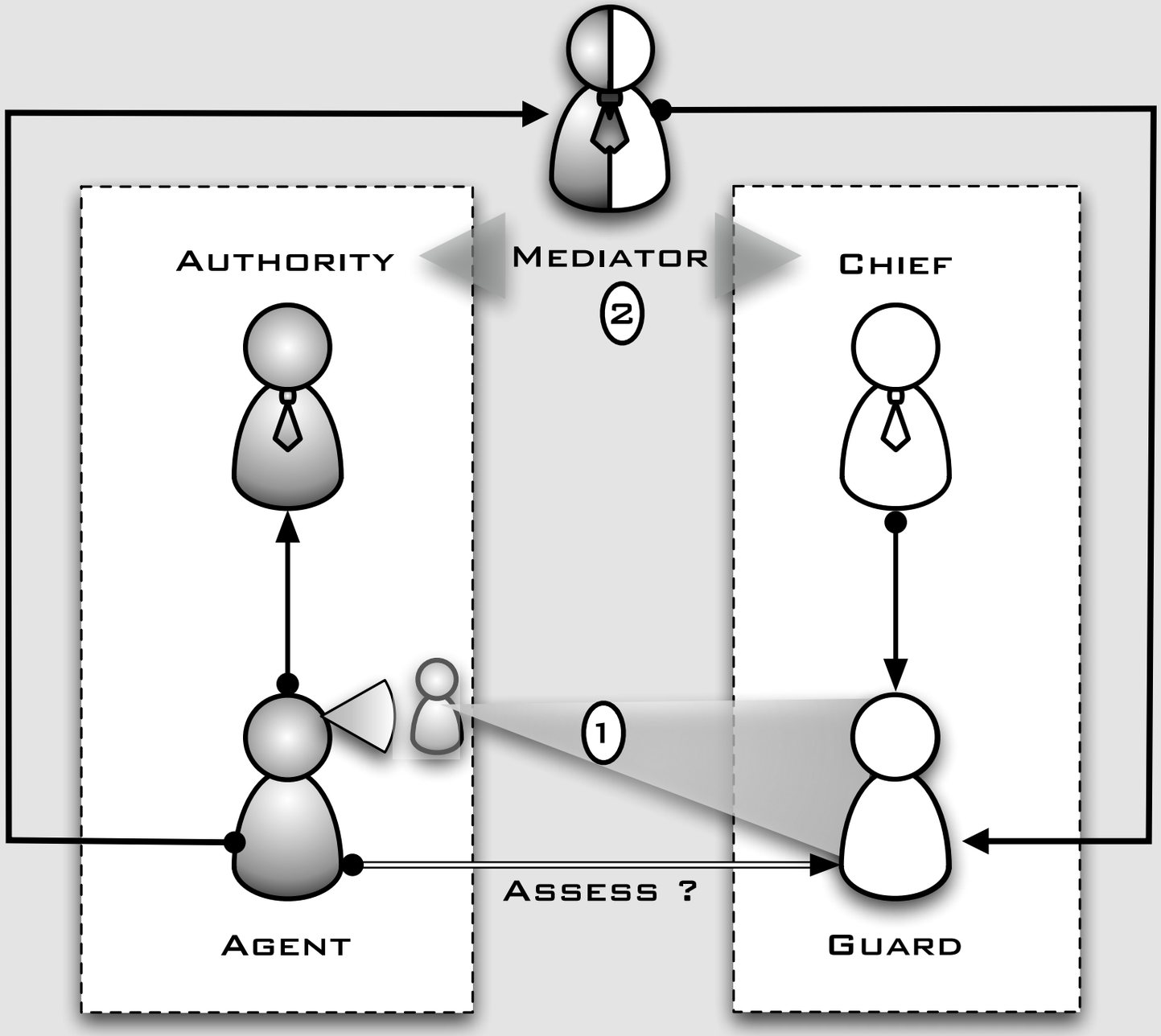
- For example:
  - ***The objective:*** an Agent has to check the accreditation of a Guard.
  - **Guards** are seen as **Agents** in the **Government trust model**.
  - The Guard will have to play the role of an Agent
    - We note: $(r_G : TM_C) \triangleright (r_A : TM_D)$

Composition Process



1- Mapping rule

Composition Process



1- Mapping rule

2- Enable mediation with third party recommenders

Trust Model Composition

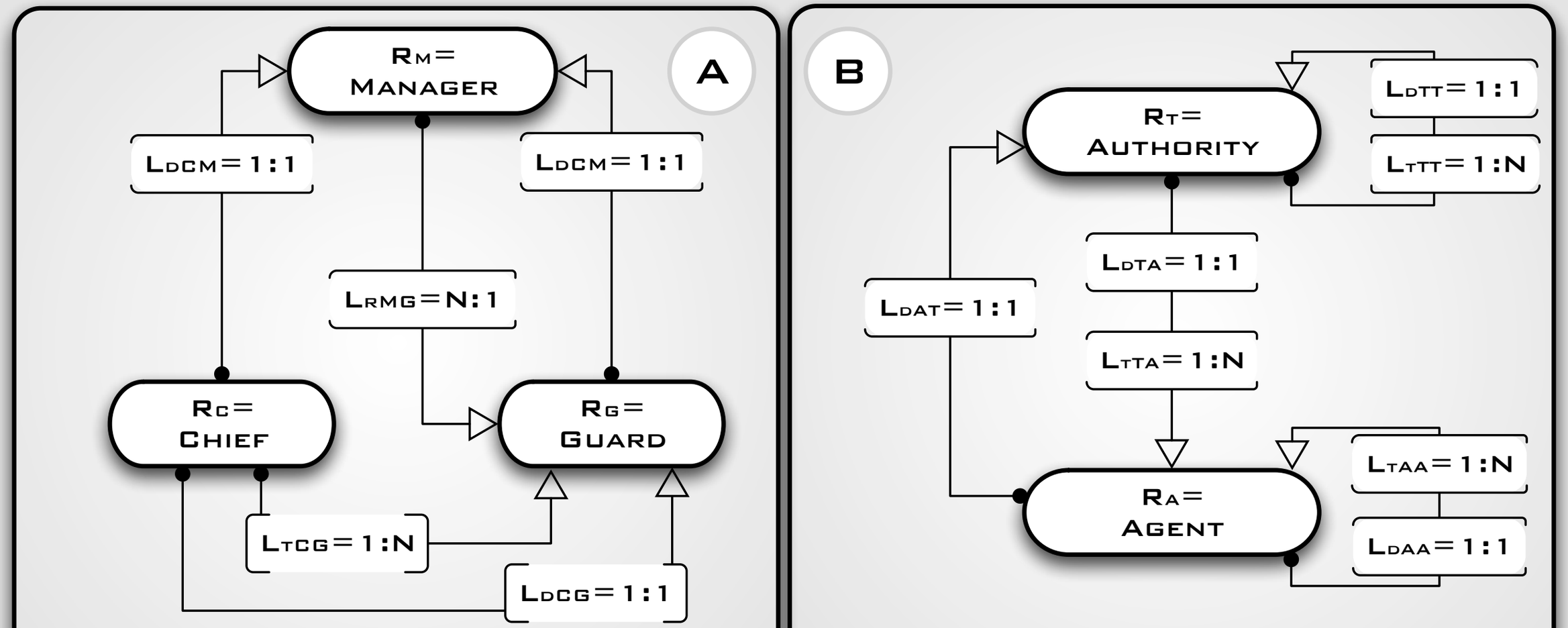
- Formally, the composition, of TM_x and TM_y, based on the set of mapping rules Ψ_{xy} is defined as follows:

$$
\begin{aligned}
TM_{xy} &= TM_x \underset{\Psi_{xy}}{\odot} TM_y \\
&= < \mathbb{R}_x, \mathbb{M}_x, \mathbb{L}_x, \mathbb{O}_x > \underset{\Psi^{xy}}{\odot} < \mathbb{R}_y, \mathbb{M}_y, \mathbb{L}_y, \mathbb{O}_y > \\
&= \left\langle \begin{array}{lcl} \mathbb{R}_{xy} & = & \mathbb{R}_x \cup \mathbb{R}_y \cup \mu\mathbb{R}_{xy} \\ \mathbb{M}_{xy} & = & \mathbb{M}_x \cup \mathbb{M}_y \\ \mathbb{L}_{xy} & = & \mathbb{L}_x^+ \cup \mathbb{L}_y^+ \\ \mathbb{O}_{xy} & = & \mathbb{O}_x^+ \cup \mathbb{O}_y^+ \cup \mu\mathbb{O}_{xy} \end{array} \right\rangle
\end{aligned}
$$

- $\mu\mathbb{R}_{xy}$ is the set of the mediator roles
- $(\mathbb{L}_x^+, \mathbb{L}_y^+), (\mathbb{O}_x^+, \mathbb{O}_y^+)$ are the extended relations and operations, respectively
- $\mu\mathbb{O}_{xy}$ is the set of mediation operations

Composition Process (Step 1)

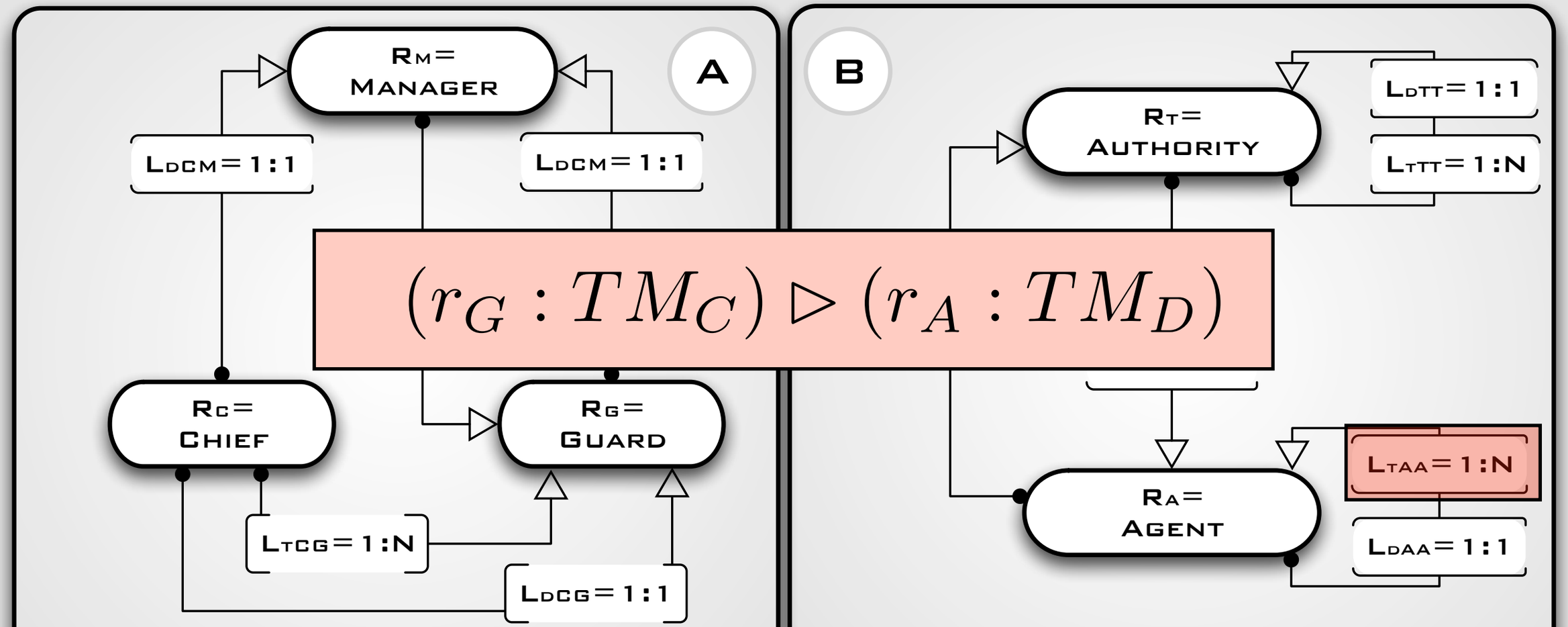
- Extend trust relations according to the mapping rule



$$l_{tAA} =< name = \text{"RemoteTrustAgent"}, ctx = \text{"Security"}, type = \text{"1:N"}, trustor = \text{"}r_A\text{"}, trustee = \text{"}r_A\text{"}, metric = \text{"}m_{Acc}\text{"} >$$

Composition Process (Step 1)

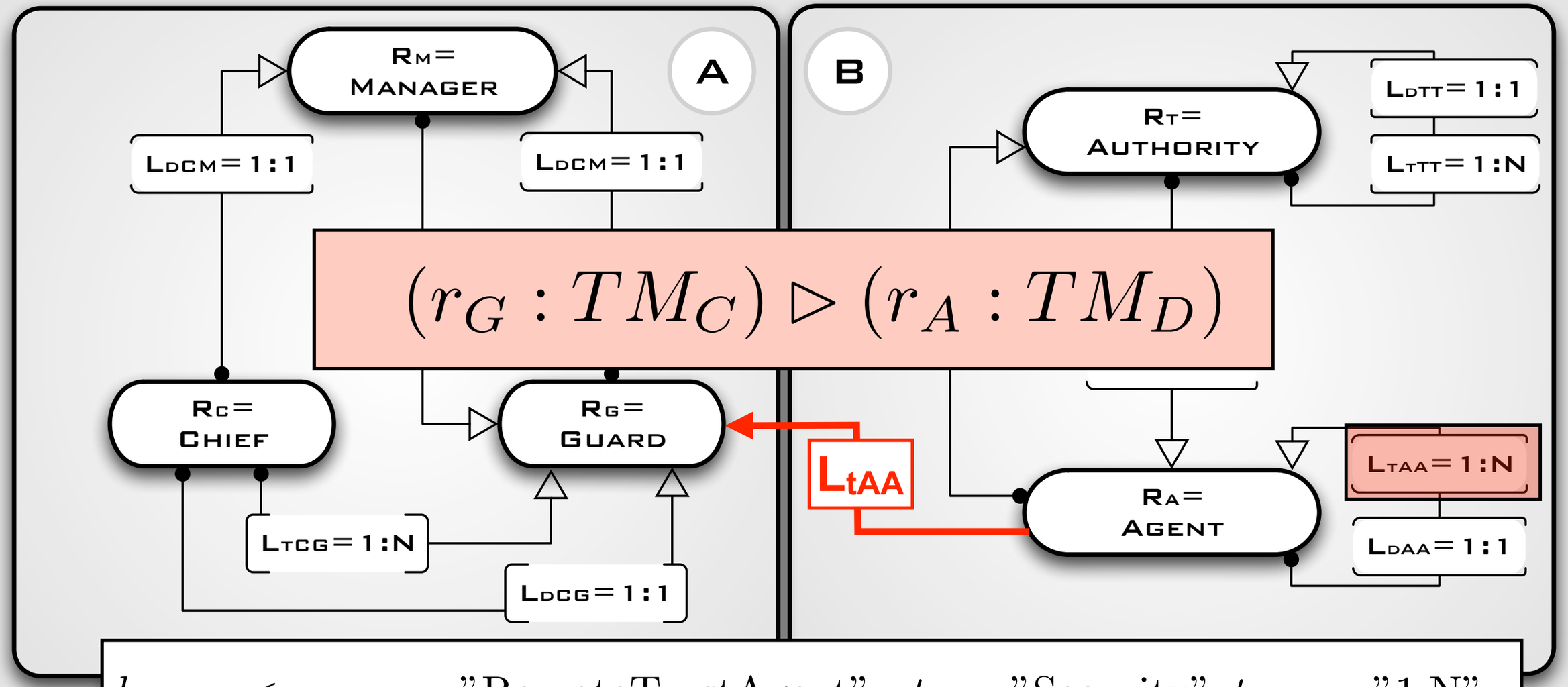
- Extend trust relations according to the mapping rule



$$l_{tAA} =< name = \text{"RemoteTrustAgent"}, ctx = \text{"Security"}, type = \text{"1:N"},$$
$$trustor = \text{"}r_A\text{"}, trustee = \text{"}r_A\text{"}, metric = \text{"}m_{Acc}\text{"} >$$

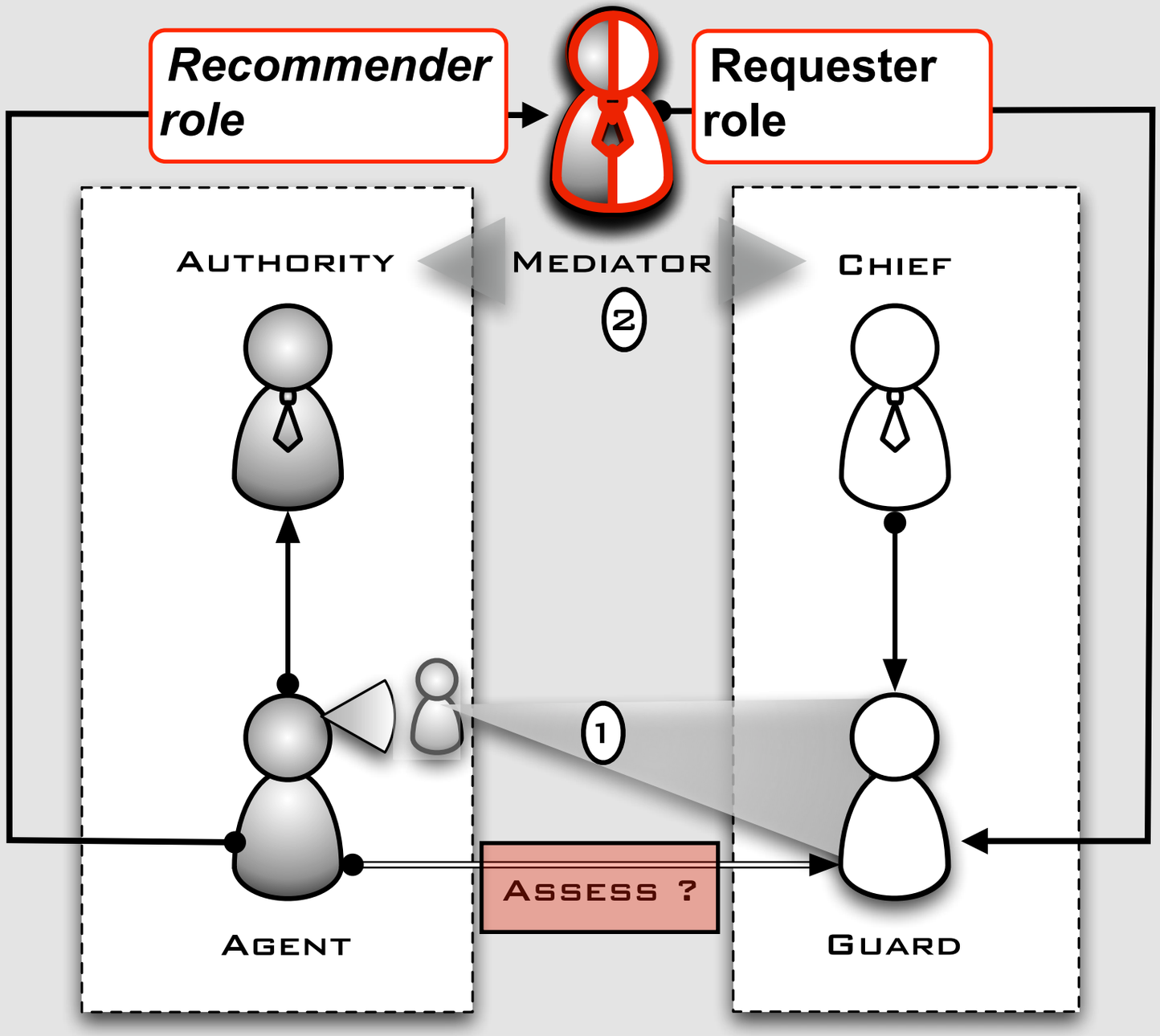
Composition Process (Step 1)

- Extend trust relations according to the mapping rule

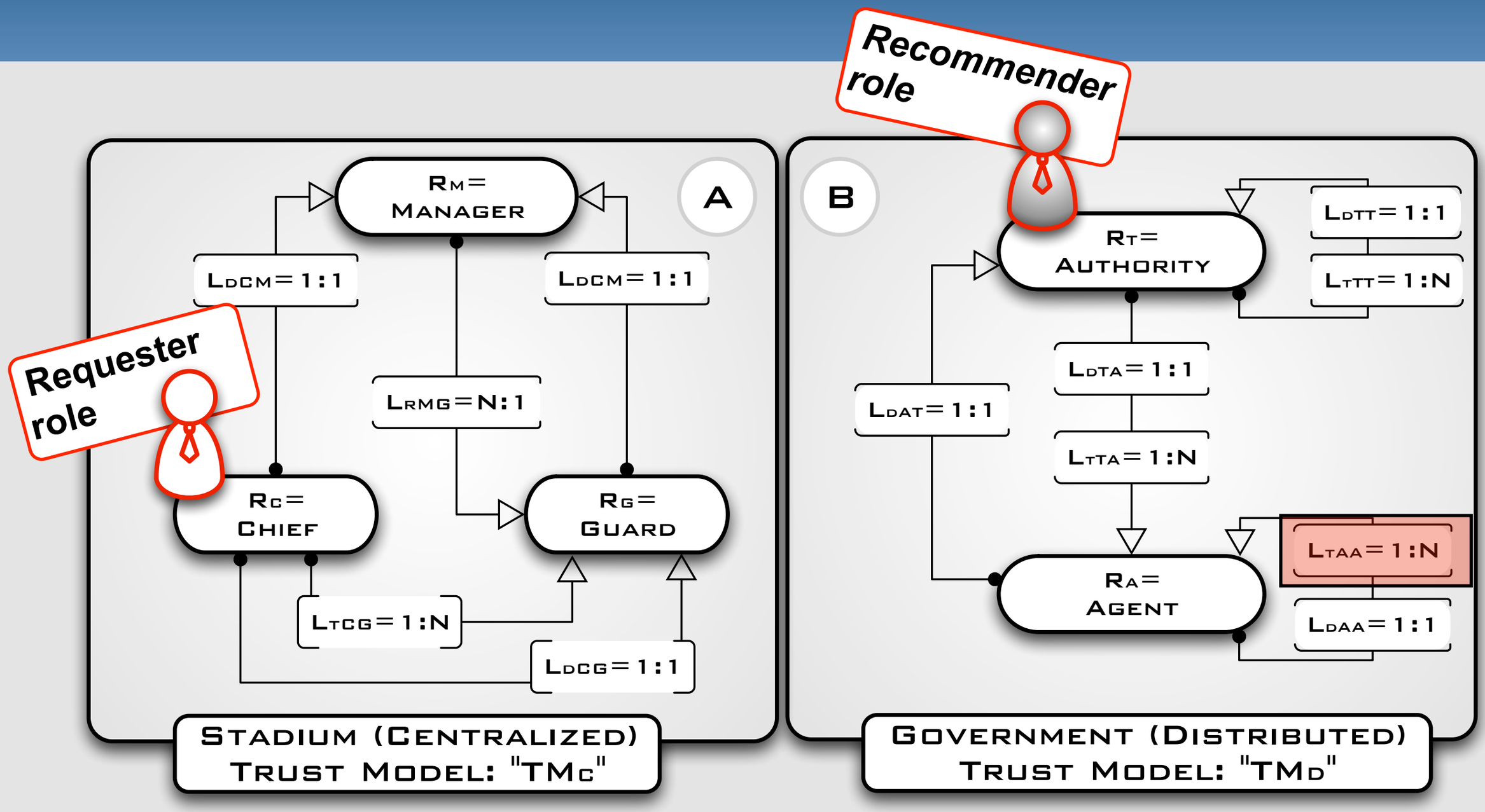


$$l_{tAA} =< name = \text{"RemoteTrustAgent"}, ctx = \text{"Security"}, type = \text{"1:N"},$$
$$trustor = \text{"}r_A\text{"}, trustee = \text{"}r_A \vee r_G\text{"}, metric = \text{"}m_{Acc}\text{"} >$$

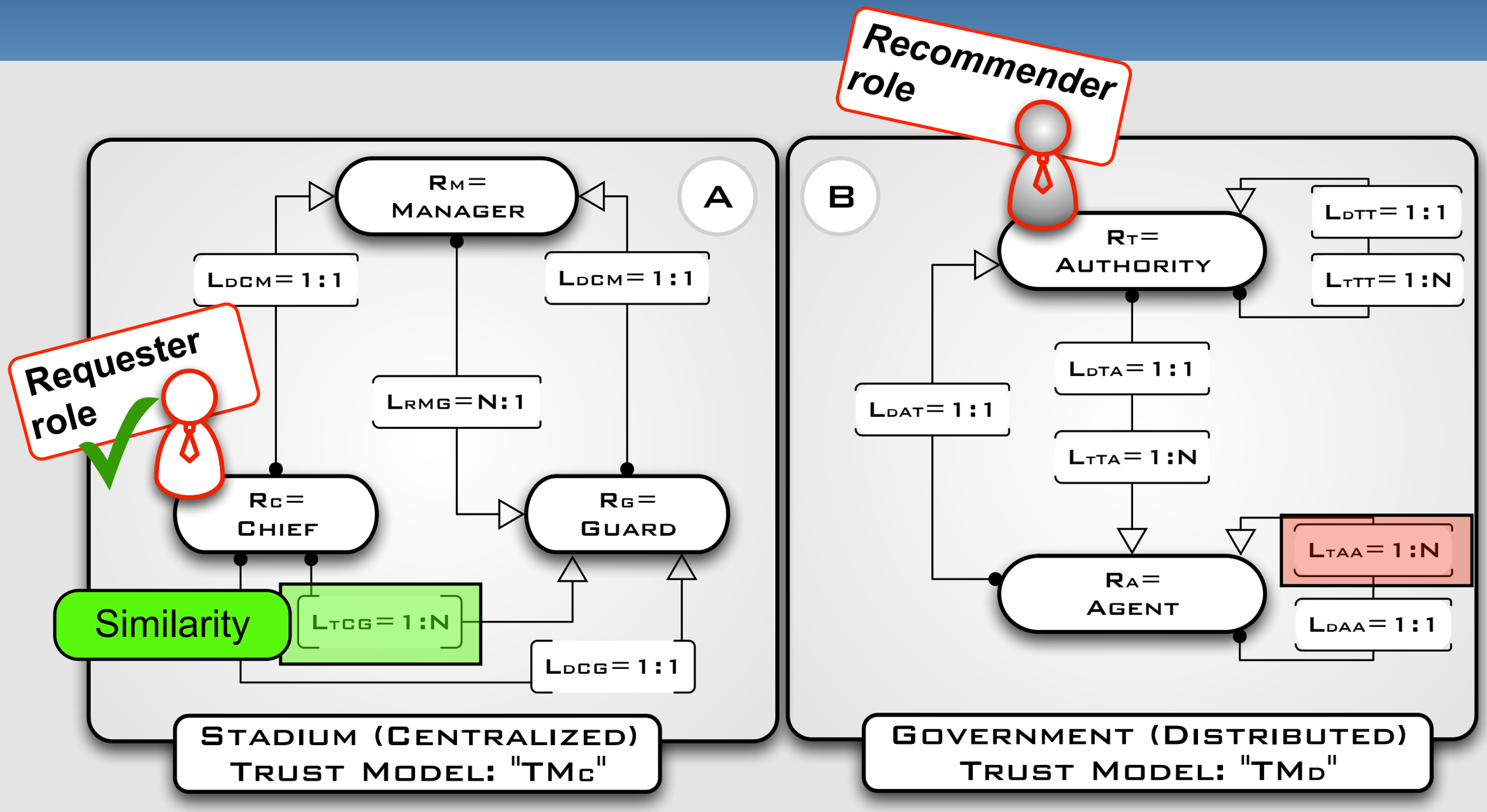
Composition Process (Step 2)



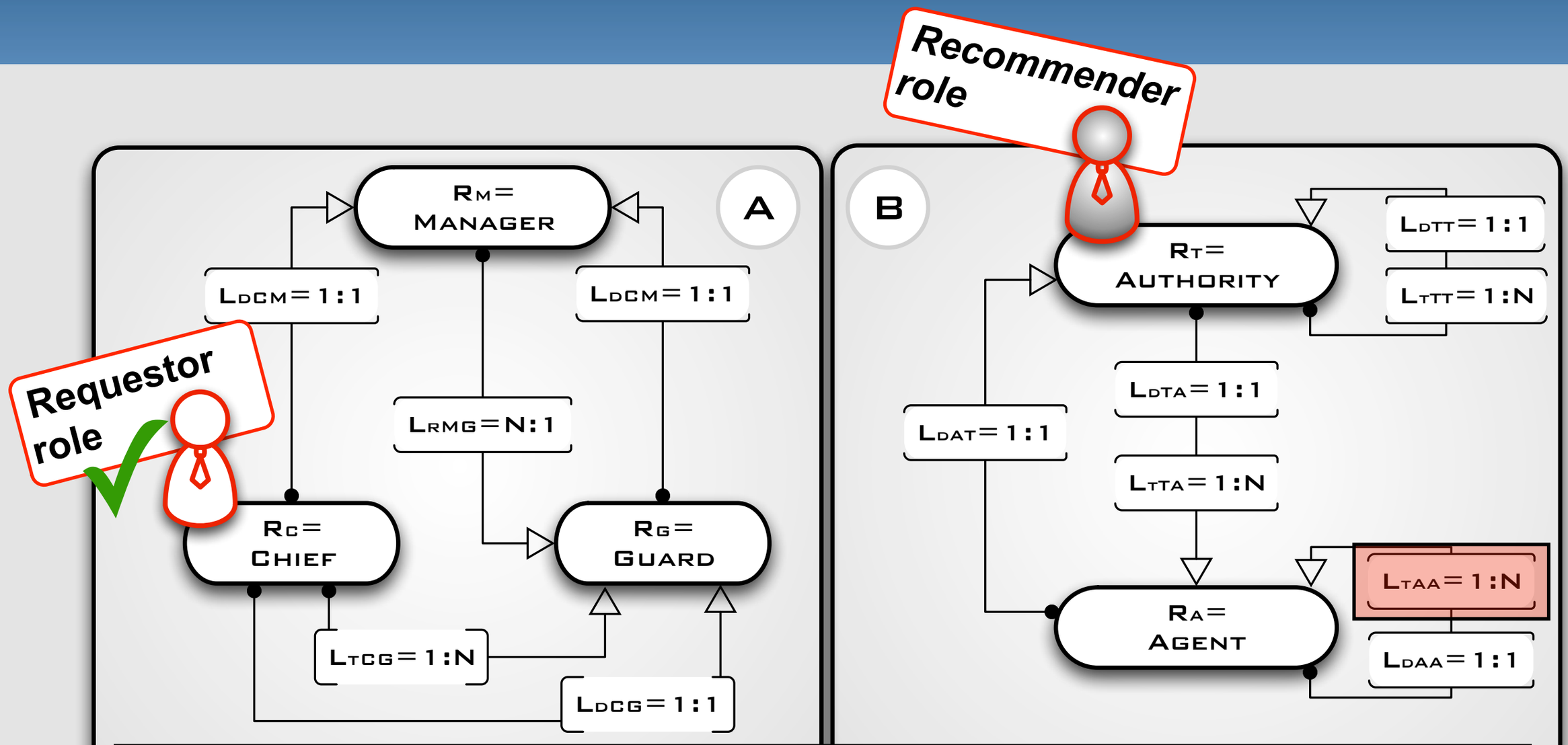
Composition Process (Step 2)



Composition Process (Step 2)



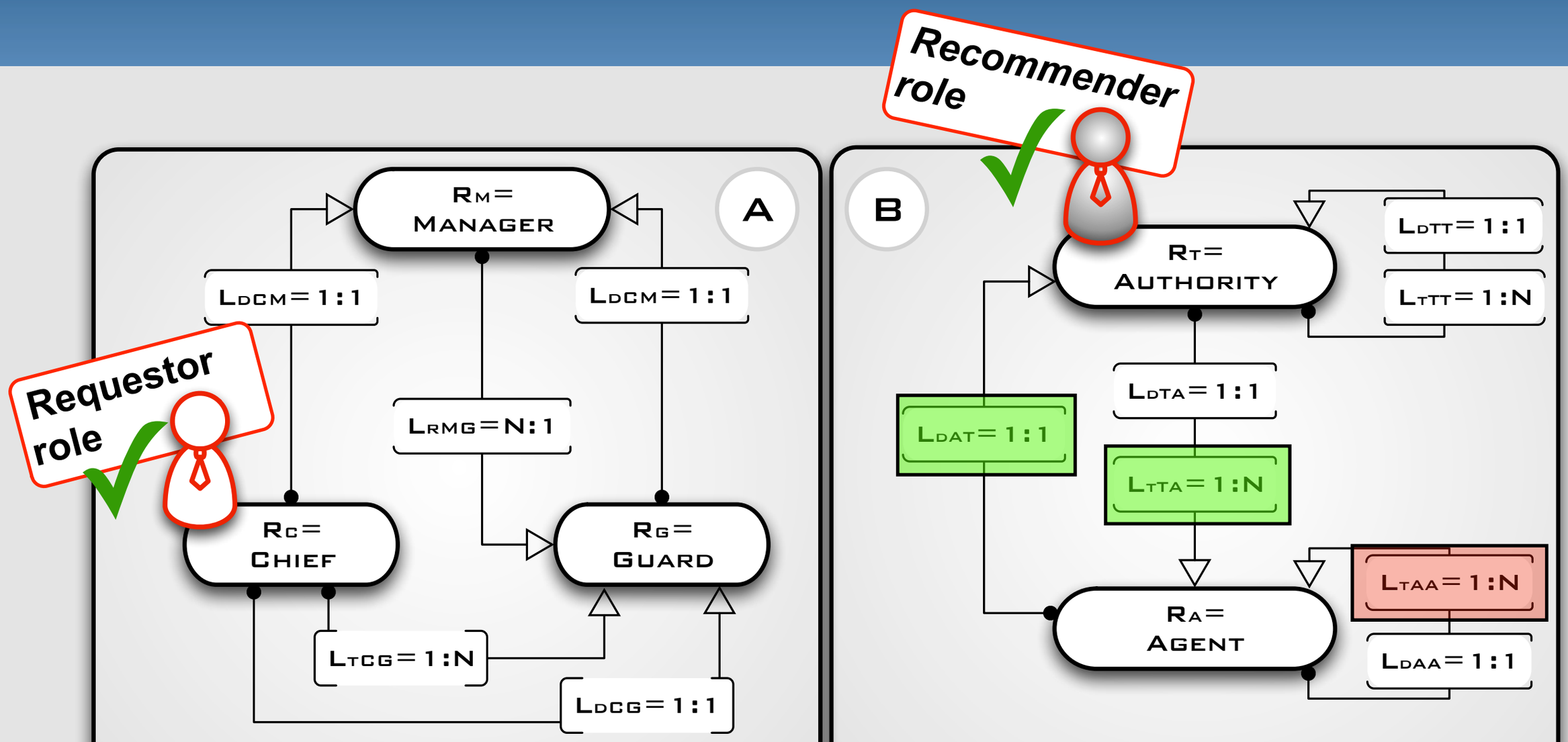
Composition Process (Step 2)



$$o_{cRP} =< name = \text{"concatRemoteAgent"}, host = r_A, type = \text{concatenation}, in = l_{dAT} \wedge l_{tTA}, out = l_{tAA} >$$

Concatenation operation

Composition Process (Step 2)



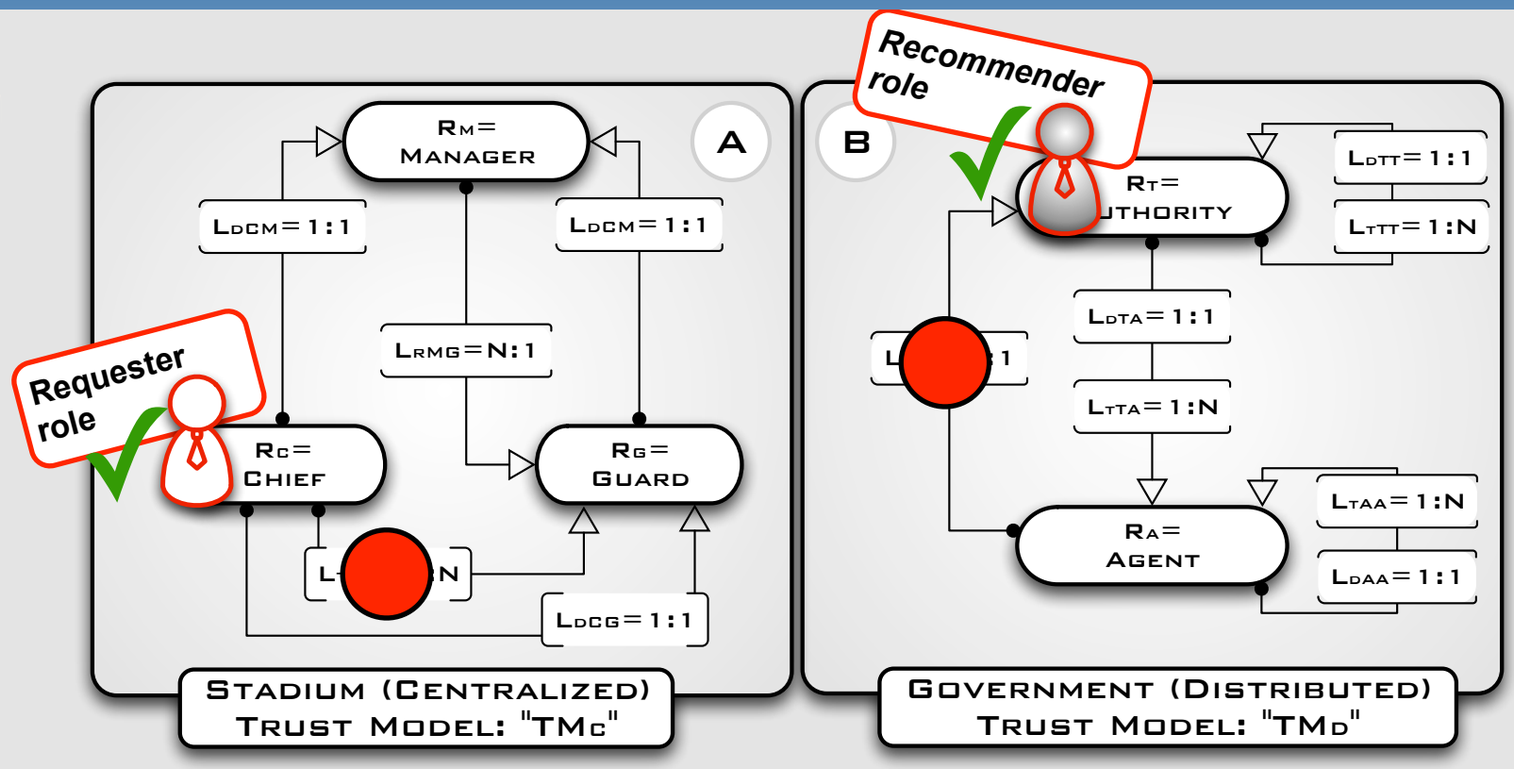
$$o_{cRP} =< name = \text{"concatRemoteAgent"}, host = r_A, type = \text{concatenation},$$
$$in = l_{dAT} \wedge l_{tTA}, out = l_{tAA} >$$

Concatenation operation

Composition Process (Step 3)

- **Extend** all **relations** that **sink** in the **recommender** role to **sink** in the **mediator** **(extend the trustee attribute)**

- **Extend** all **relations** that **start** from the **requestor** to also **start** from the **mediator** **(extend the trustor attribute)**



- Extend all the **operations** that are performed by the **requestor** and the **recommender** to the **mediator (extend the host attribute)**

- Enhance the **mediator** with **mediation operations** to **link operations** and to **translate metrics**' value **(extend the call attribute)**

Trust Conclusion

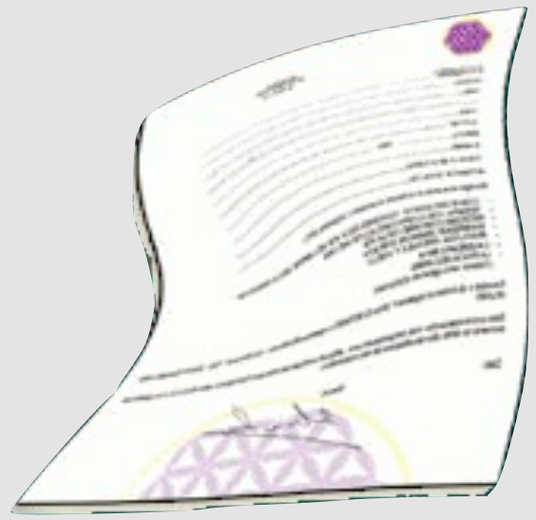
- We defined a meta-model and illustrate its expressivity over a centralized and a distributed scenario
- We presented an automated trust model composition with mediation
- Heterogeneous networked systems are able to assess each other
- Security-by-Contract can be extended with Trust for guaranteeing more efficient run-time security

Security Outline

- Security-by-Contract (SxC)
  - Contract definition
  - Policy definition
  - SxC architecture
- Extending SxC with Trust (SxCxT)
  - Trust management module
  - Contract monitoring
  - SxCxT architecture
- Application to the CONNECT world
- Conclusion

Contract

A ***contract*** is a formal complete and correct specification of the behavior of the application for what concerns relevant security actions (e.g., Virtual Machine API call, Operating System Calls). It defines a subset of the traces of all possible security actions.



Policy

A ***policy*** is a formal complete specification of the acceptable behavior of applications to be executed on the platform for what concerns relevant security actions (e.g., Virtual Machine API call, Operating System Calls).

It defines a subset of the traces of all possible security actions.

Security-By-Contract (SxC)

Security-By-Contract (SxC)

The main goal is to ensure that the application satisfies the policy!

Security-By-Contract (SxC)

The main goal is to ensure that the application satisfies the policy!

Two ways:

Security-By-Contract (SxC)

The main goal is to ensure that the application satisfies the policy!

Two ways:

1. The application **satisfies** the contract and the contract adheres to the policy. Then the application can be safely executed with further controls.

Security-By-Contract (SxC)

The main goal is to ensure that the application satisfies the policy!

Two ways:

1.The application **satisfies** the contract and the contract adheres to the policy. Then the application can be safely executed with further controls.

- How to check that the code adheres to the contract and the contract to the policy: (Abstract interpretation, Model checking, Trust)

Security-By-Contract (SxC)

The main goal is to ensure that the application satisfies the policy!

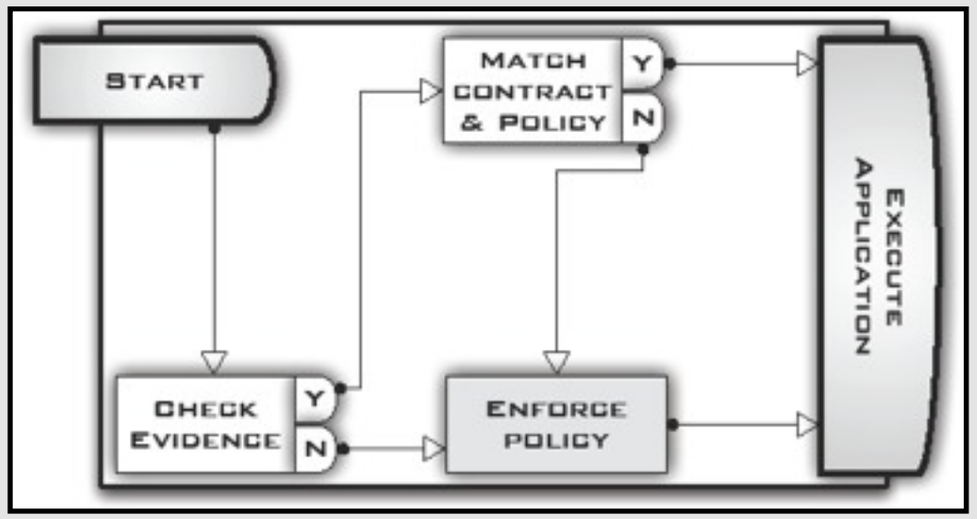
Two ways:

1. The application **satisfies** the contract and the contract adheres to the policy. Then the application can be safely executed with further controls.
    - How to check that the code adheres to the contract and the contract to the policy: (Abstract interpretation, Model checking, Trust)

2. The application **does not satisfy** the contract, or the contract does not adhere to the policy, the policy is directly enforced at run-time.

SxC Architecture

Two approaches:



- Prior midlet execution at **deployment time** (Verification of code/policy compliance through contracts)
- During midlet execution (**run-time** enforcement)

Verification at Deployment Time

Basically, we check on device two facts:

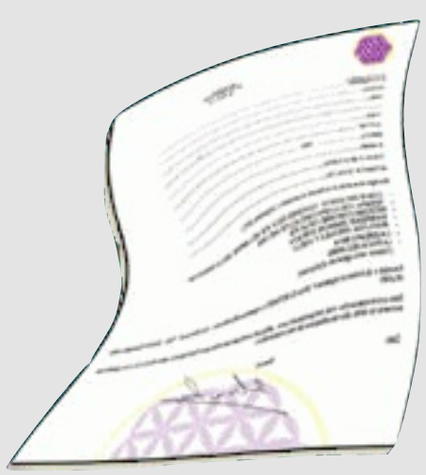
- The code is compliant with the contract
  - This can be done in several ways
    - *Proof carrying code, trust on code/contract*
      - signer
- The contract is compliant with the local policy
  - We check that one process simulates the other by using symbolic simulation techniques

Contract-Policy Matching

The idea underlying ***contract-policy matching*** is that any sequence of security relevant actions allowed by the contract is also allowed by the policy.

Contract-Policy Matching

The idea underlying ***contract-policy matching*** is that any sequence of security relevant actions allowed by the contract is also allowed by the policy.



matches

Contract-Policy Simulation Matching

We use **simulation relation** for **contract-policy matching**.

A **contract** transition system **is simulated** by a **policy** one, if, for each transition corresponding to a certain security action of the contract (and so possibly performed by the program), the policy has a similar transition and resulting contract system is yet simulated by the resulting policy one.

Security-By-Contract

Two ways:

1. The applet **satisfies** the contract and the contract adheres to the policy. Then the applet can be safely executed;
2. The applet **does not satisfy** the contract, or the contract does not adhere to the policy, the policy is directly enforced at run time.

One crucial point is the checking of the satisfaction that the applet satisfies the contract:

- Abstract interpretation
- Model checking
- Simply trust on the a third party that certify that the applet satisfies the contract (in the simplest case the certifier is actually the code provider).

Why SxCxT?

In the SxC paradigm, the mobile code is run if **its origin is trusted**.

SxCxT extends it by adding a component for the **contract monitoring** that allows us to check if ***the execution of the application adheres to its contract*** and, according to the answer, we **update the level of trust** of the provider.

How SxCxT Extends SxC

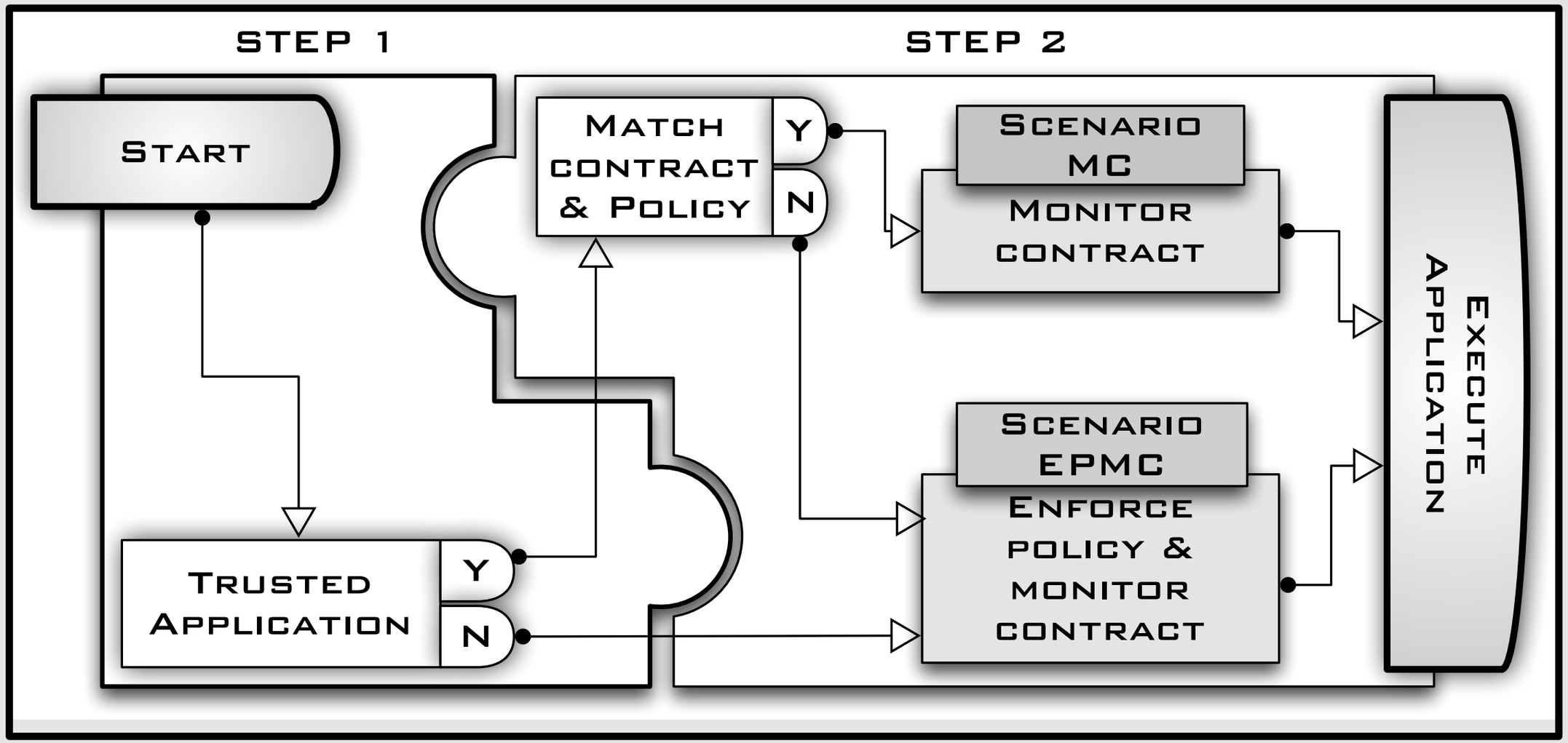
The S×C×T paradigm extends the S×C in two phases:

- at **deploy-time** by setting the monitoring state
- at **run-time** by applying the contract monitoring procedure for adjusting the provider trust level

The SxCxT has two new modules:

- a **run-time contract monitoring** for allowing to manage applications trust level and
- a **trust management** mechanism able to guarantee that a mobile application can be safely executed on mobile devices, according to trust recommendations.

Security-by-Contract with Trust Architecture



SxCxT Steps (1)

Step 1- Trust Assessment: Once a mobile application is downloaded on the mobile device, before executing it, the trust module decides if it is trusted or not according to a fixed trust threshold.

Step 2- Contract Driven Deployment: According to this trust measure, the security module defines if just monitoring the contract or both enforce the policy and monitoring the contract.

SxCxT steps (2)

Step 3 –Contract Monitoring vs Policy Enforcement Scenarios: Depending on the chosen scenario the security module is in charge to monitor either the policy or the contract and save the execution traces (logs).

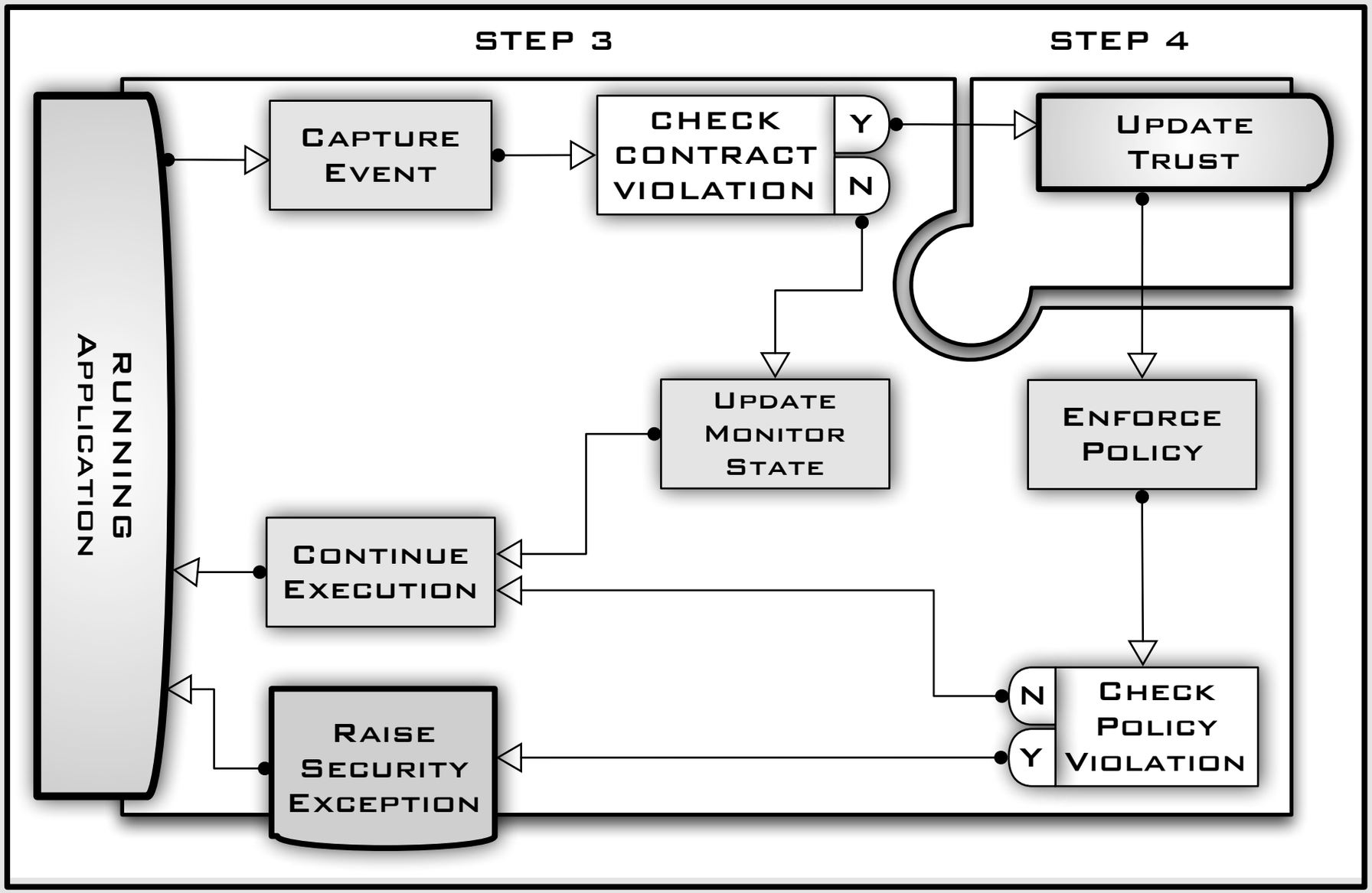
Step 4– Trust Feedback Inference: Finally, the trust module parses the S×C×T produced logs and infers trust feedback.

MC Scenario (1)

Scenario MC: The contract satisfies the policy.

- Only **the contract is monitored**, i.e., we check is he execution adheres to the contract
  - under these conditions, **contract adherence** also **implies policy compliance**.
- If **no violation** is detected then the application worked as expected.
- If **there is a violation**, this means that a **trusted party** provided us with a **fake contract**.
  - Our framework reacts to this event by **updating the level of trust** of the provider and switching to the policy enforcement modality.

MC Scenario (2)



EPMC Scenario (1)

Scenario EPMC: The contract does not satisfy the policy.

policy enforcement is turned on

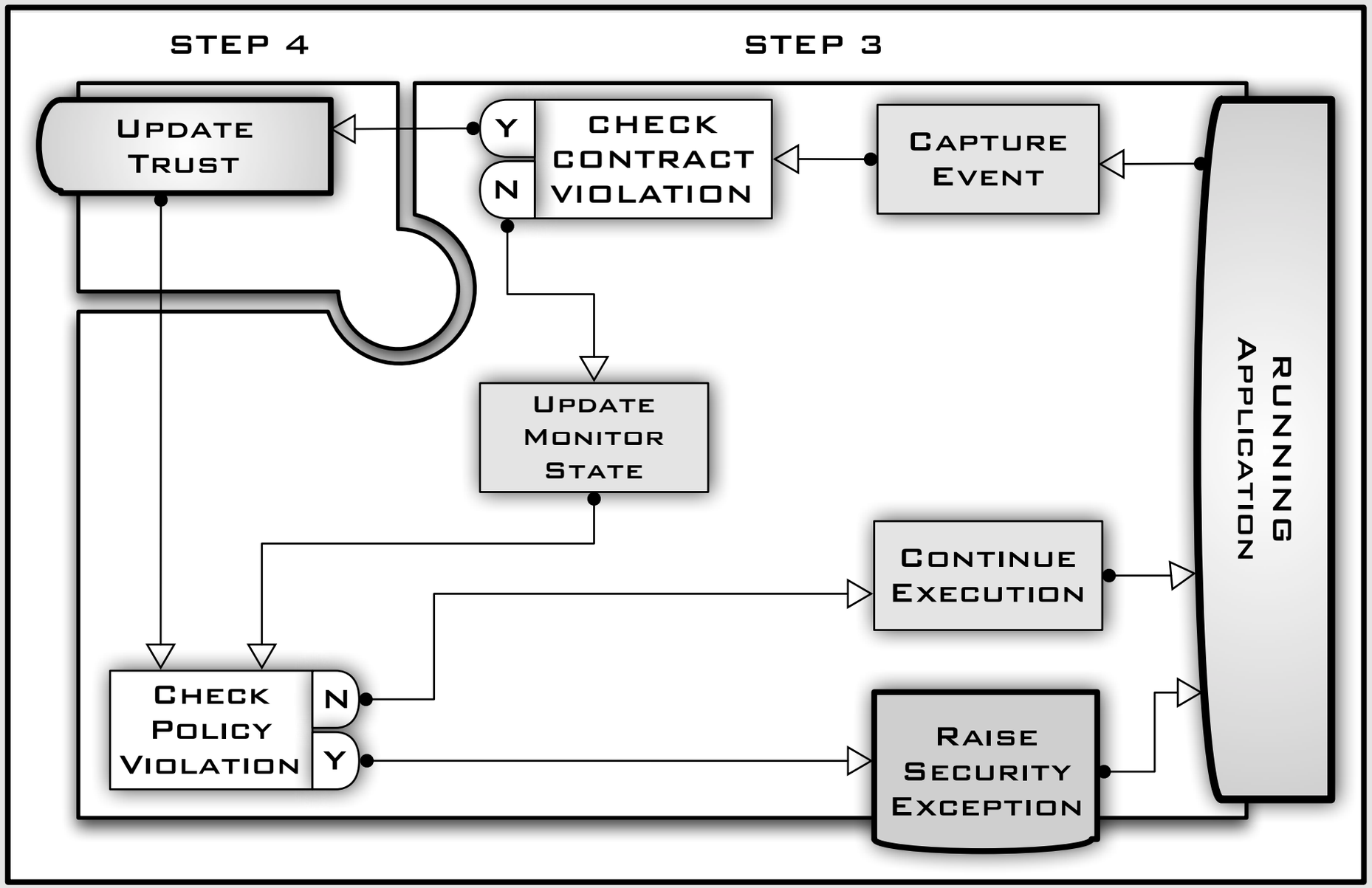
Monitoring contract is turned on

It provides a useful **feedback** for better tuning the trust vector.

Our framework also allows for a **mixed monitoring and enforcement configuration**.

This configuration is activated on a statistical base.

EPMC Scenario (2)



The CONNECT World

Networked Systems are the end-users in our network.

Enablers are the entities that receive requests from users. They store all information for creating/ choosing an appropriate CONNECTor for allowing the communication among different networked system.

CONNECTors are the final product of our framework. CONNECTors allow different devices used by different networked systems to communicate with each other even if they use different interface communication protocols.

Contract in CONNECT

A *contract* (in CONNECT) represents the correct specification of the behavior of the CONNECTor provided by the enabler for allowing the communication

Policy in CONNECT

A *policy* (in CONNECT) is set a local policy of each networked system. It declares what each NS aspects from the CONNECTor.

Security in CONNECT According to the CONNECTor Life-cycle

	CONNECT Infrastructure	Between CONNECT and NSs	CONNECTed system
Synthesis Time	i)All Enablers are safe; ii)Some Enabler is malicious	i)The NSs are safe; ii) At least one NS is malicious	
Deployment Time		i)The Nss are safe; ii) At least one NSs is malicious	
Execution Time	i)All Enablers are safe; ii)Some Enabler could be malicious	i)The Nss are safe; ii) At least one NSs is malicious	i)Each NS could be a malicious agent ii) CONNECT does not trust NSs

Security in CONNECT According to the CONNECTor Life-cycle

	CONNECT Infrastructure	Between CONNECT and NSs	CONNECTed system
Synthesis Time	i)All Enablers are safe; ii)Some Enabler is malicious	i)The NSs are safe; ii) At least one NS is malicious	
Deployment Time		i)The Nss are safe; ii) At least one NSs is malicious	
Execution Time	i)All Enablers are safe; ii)Some Enabler could be malicious	i)The Nss are safe; ii) At least one NSs is malicious	i)Each NS could be a malicious agent ii) CONNECT does not trust NSs

Security between the Infrastructure and the NSs Layer

Threat Model:

Each Networked System involved could be a malicious agent. NSs trust the CONNECT framework but they do not trust each other. The CONNECT framework does not trust NSs.

Security between the Infrastructure and the NSs Layer

- Centralized: In this case the CONNECTor and the enabler, that runs the connector, can be considered as the same thing form the security point of view.
- Security policies are set on the running platform, i.e. on the enabler that run the CONNECTor.
- The security policies of the NSs cannot be enforced

Security between the Infrastructure and the NSs Layer

Distributed

1)if the CONNECTor run only on the NSs there is no security constraints between the infrastructure and the NS layer because the CONNECTor is already deployed and it is out of the control of the infrastructure.

2) if part the connector is deployed on one of the enabler of the infrastructure we refer to the centralized case

Each Networked System Runs its Own Part of the CONNECTor

Two possible scenarios arise:

- the NSs **are provided** of the SxCxT mechanism.

- the NSs **are not provided** of the SxCxT. In this case we are in the same case of the centralized scenario

NSs run SxCxT (1)

The SxCxT paradigm is applied for guaranteeing at run-time that:

- local private policies, that are not shared neither with the CONNECT infrastructure, are satisfied.

Example of local private policies are private policies regarding, for instance the GPS coordinates of he NS itself, the right for accessing to the private data stored on the device of the NS, and so on.

NSs run SxCxT (2)

- none NS tries to attack the other NS by manipulating the part of the CONNECTor running on it.

One of the two NS is a malicious agent. When it receives the part of CONNECTor that it is in charge to run, the NS manipulates it in order to attack the other NS during the communication.

Conclusion

We apply of the Security-by-Contract-with-Trust framework to CONNECT world. Indeed we show a strategy for establishing a secure communication among components of the Network by guaranteeing that they are connected through the usage of a secure CONNECTor