How to generate security cameras: Towards defence generation for socio-technical systems

predict prioritise prevent
TRESPASS

UNIVERSITÉ DU LUXEMBOURG

ADT2P

Olga Gadyatskaya (SnT, University of Luxembourg)
olga.gadyatskaya@uni.lu

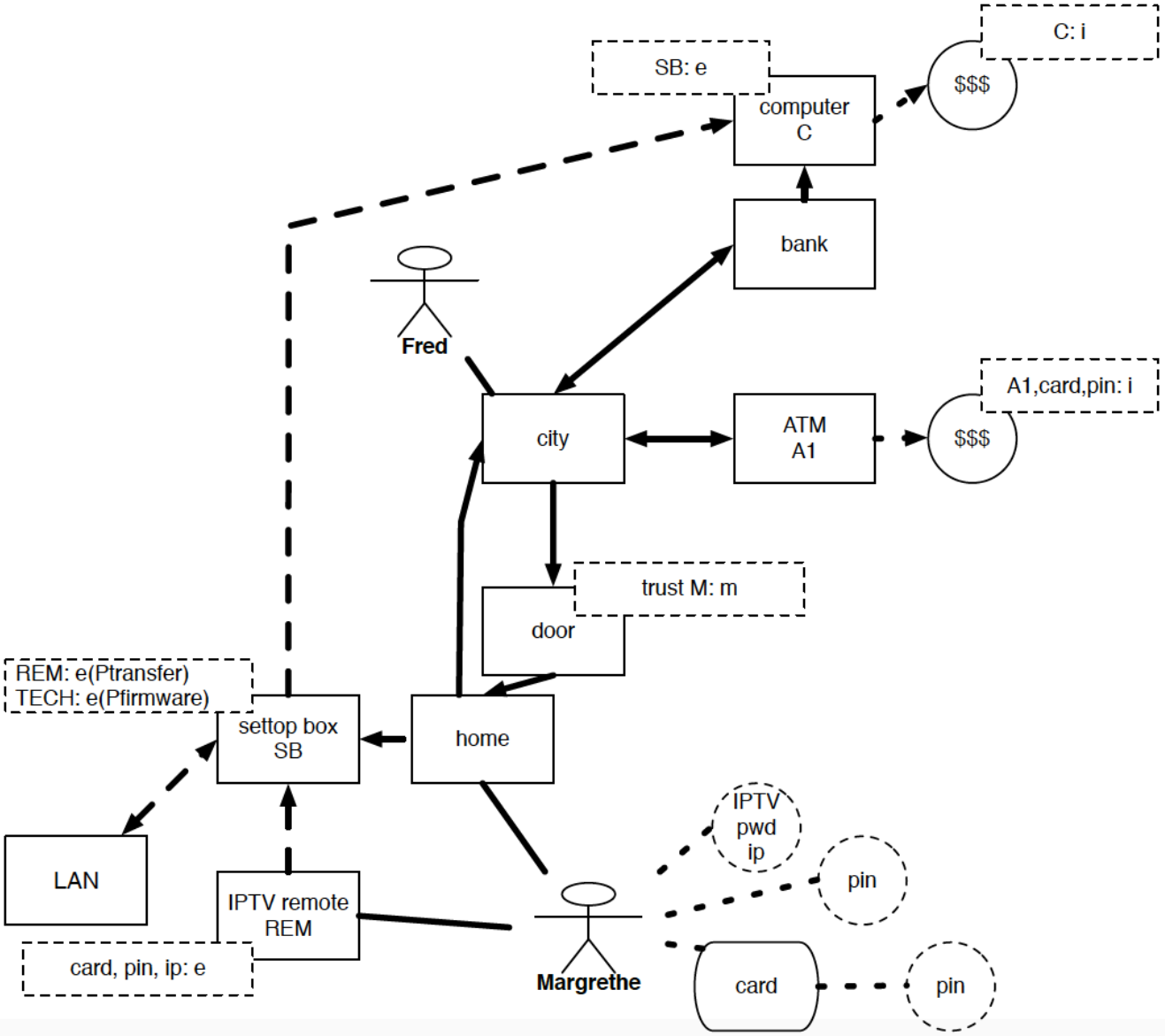
Agenda

- Socio-technical models and attack generation
- Challenges for countermeasure generation
- Attack-defence model generated from socio-technical model
- How to select more countermeasures
- Challenges ahead

Socio-technical system models

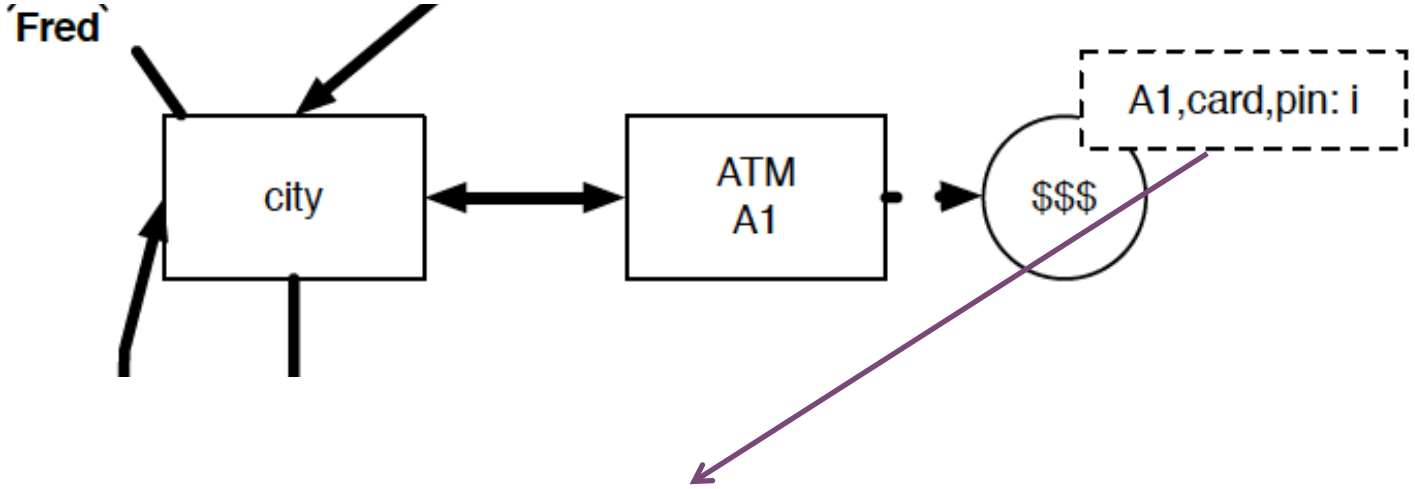
- A model that combines a snapshot of infrastructure with models of agents acting in this infrastructure

+ Example



An instance of a system model designed in the TREsPASS notation

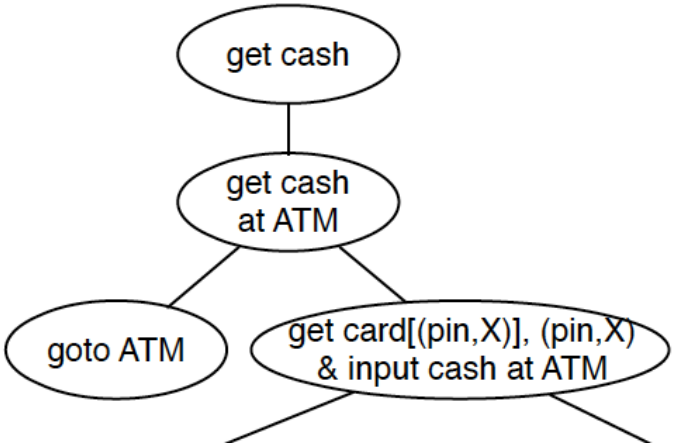
Security controls in the model



Money $$$ can be accessed from the ATM A1 with card and PIN.

Automated attack generation

- A socio-technical model → an attack model [Ivanova et al. 2015]
  - automatically
  - complete wrt the socio-technical model
  - reachability-based



An example of a generated attack tree [Ivanova et al. 2015]

Automated generation of countermeasures: challenges

- <easy> Which format for countermeasure representation?
  - Attack-countermeasure trees, attack-defence trees, defence trees, etc.
- <hard> Generated countermeasures are limited by the socio-technical model itself
  - If the model represents only access control policies – only those can be generated automatically

Problem

- **Automated countermeasure generation**
  - How to generate defences automatically {in an optimal way}
  - How to introduce more countermeasures
  - How to trace the generated countermeasures back to the ST model and maintain the traceability through model evolution

- Solution
  - Maintain an ***attack-defence model*** together with the socio-technical system model

Attack-defence model

- The desired attack-defence model should:
  - incorporate existing countermeasures (access control policies)
  - allow to add new defences and consistently maintain traceability with the socio-technical model
  - allow to perform computations and select optimal defence scenarios

- Attack-defence trees [Kordy et al. 2014] is a suitable notation to maintain the attacker and the defender views simultaneously

Simplified attack-defence model

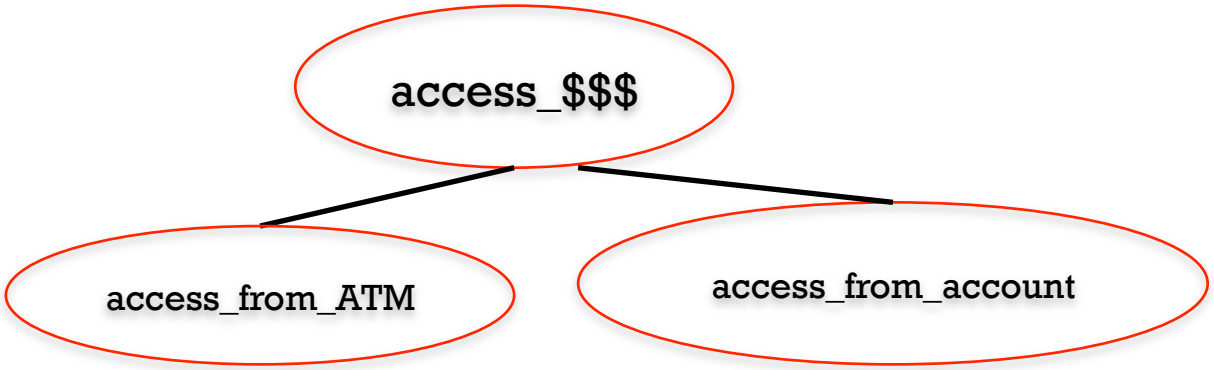
- Given a socio-technical model <*N,E*>
  - *N* is a set of items in the model
    - N_i – infrastructure locations
    - N_a – actor locations
    - N_o – object locations
  - E is a set of directed edges among the items
- P is a set of access control policies defined in the model
  - d_n is a local policy that guards access to item n
  - each element in d_n is <*Cred, atLocation, EM*> where
    - *Cred* is a set of credentials required
    - *atLocation* is the location where policy is applied
    - *EM* is an enforcement mechanism in the model

Bundles

- For each element *n* of the model we generate an **attack-defence bundle** *access_n*
  - A bundle succinctly represents an attack where an attacker gets access to *n*
    - Any attacker
  - It comprises the attack vectors available in the model and the defences offered by the enforcement mechanisms for local policies

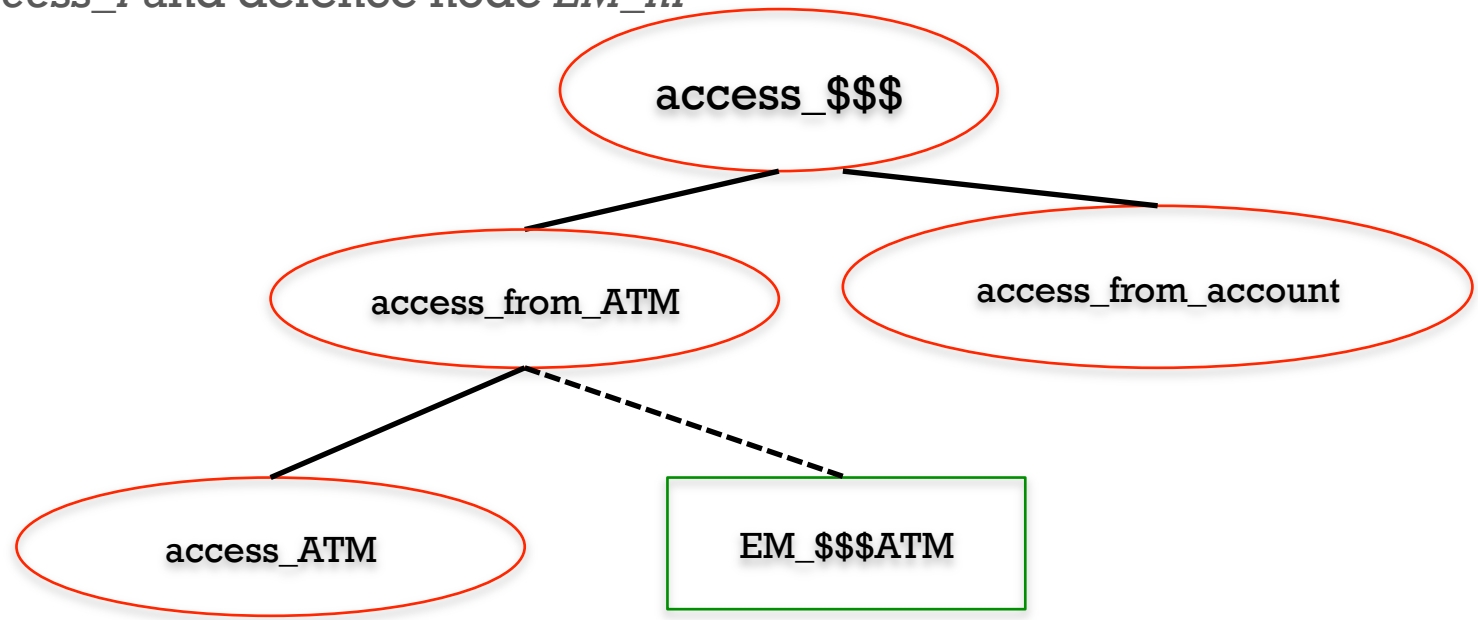
Structure of attack-defence bundles I

- Root node: *access_n*
- *n* can be accessed from any adjacent location in the model
  - *access_n* is OR-decomposed into a collection of nodes *access_from_ni*



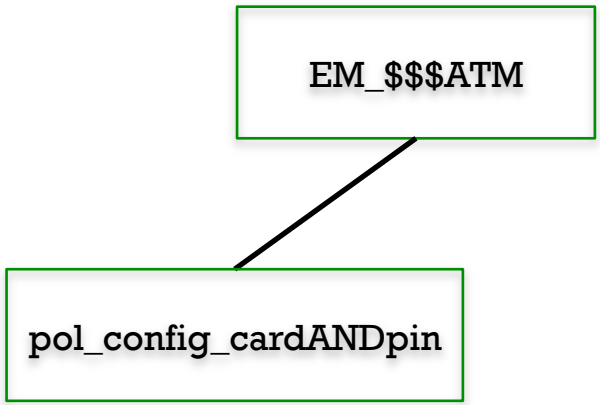
Structure of attack-defence bundles II

- To attack from some adjacent location the attacker needs to get to that location and circumvent the access control policies checks there
  - Bundles *access_from_ni* are decomposed into attack node *access_i* and defence node *EM_ni*



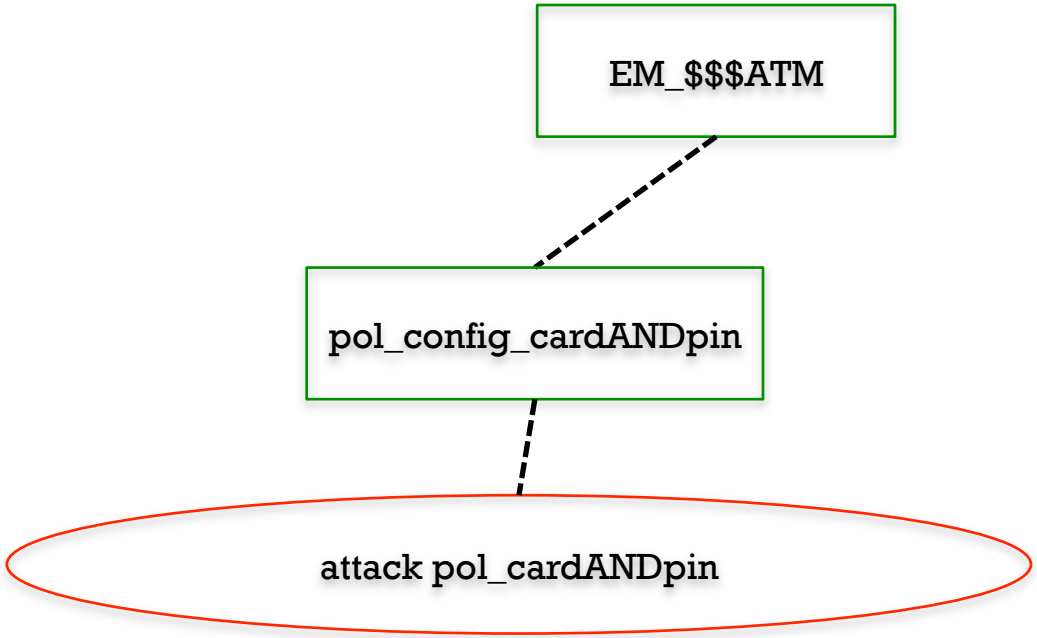
Defence nodes decomposition

- Enforcement mechanism can comprise several valid policy configurations
  - defence node *EM_ni* is AND-decomposed into nodes *pol_config_pk* each local policy configuration that guards access to *n* from *i*



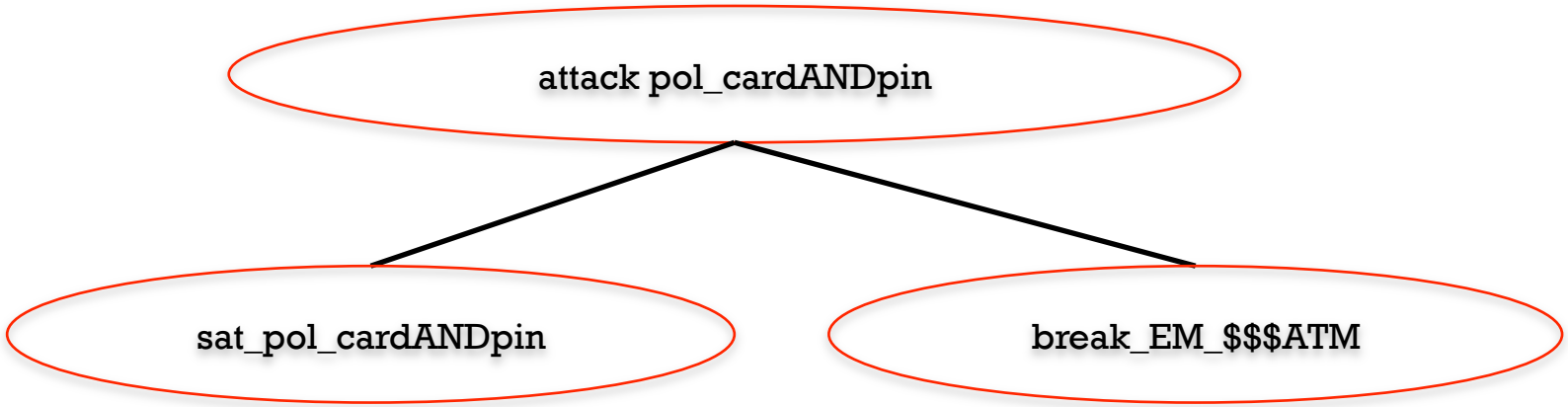
Attacking enforcement mechanisms I

- To overcome the defensive mechanism in place, the attacker needs to circumvent any of individual policy configurations



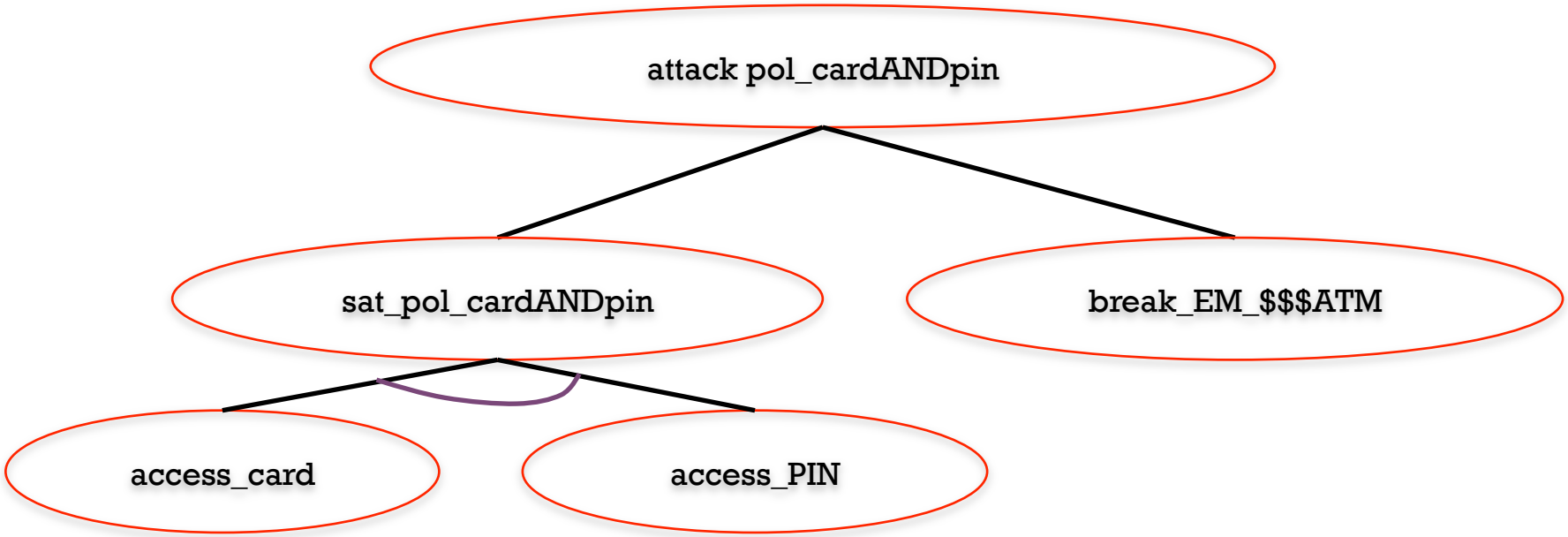
Attacking enforcement mechanisms II

- The attacker can circumvent the enforcement mechanism by satisfying the policy (collecting all credentials) or by breaking the enforcement mechanism
  - Node *attack_pol_pm* is OR-decomposed into attack nodes *sat_pol_pm* and *break_em_ni*



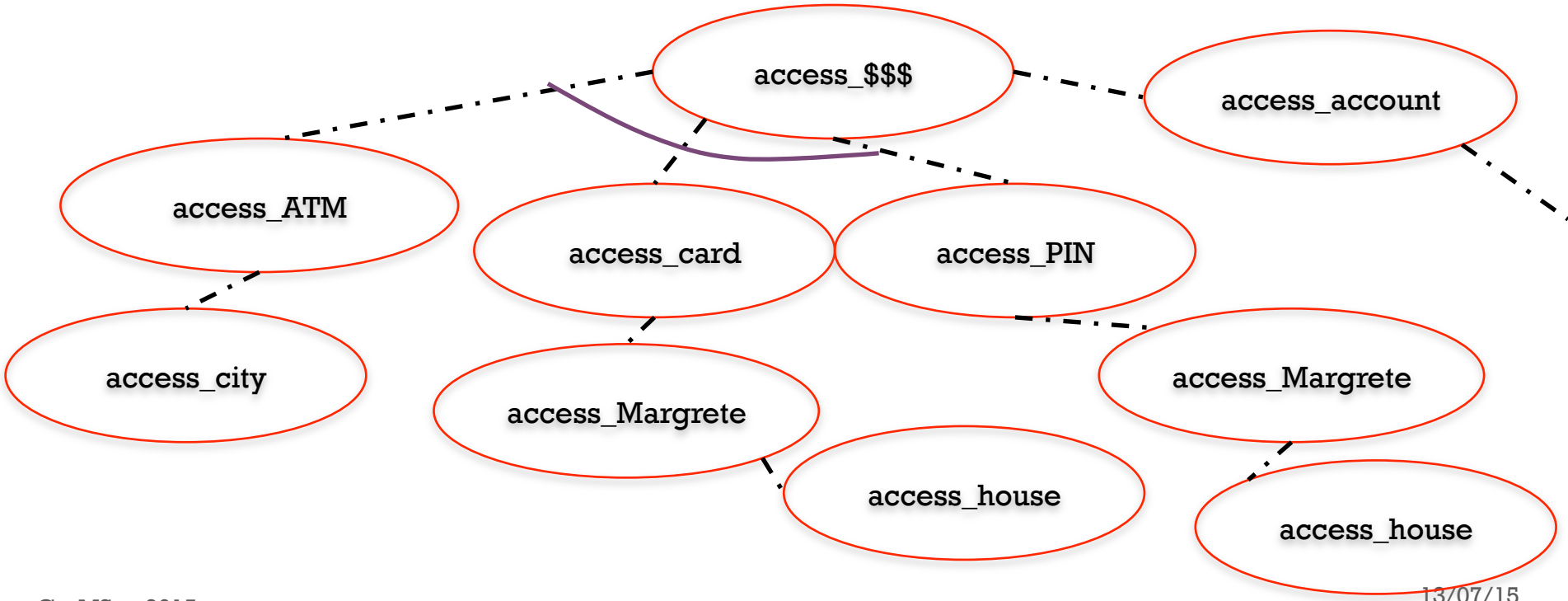
Satisfying policies

- Policy can be satisfied if all credentials needed are collected:
  - Attack node *sat_pol_pm* is AND-decomposed into attack nodes *access_credr*



Attack-defence tree synthesis from bundles I

- Attack node *access_n* is a basic building block
  - Bundles can be put together to form attack-defence trees
  - Issue: loops



Attack-defence tree synthesis from bundles II

- Solution:

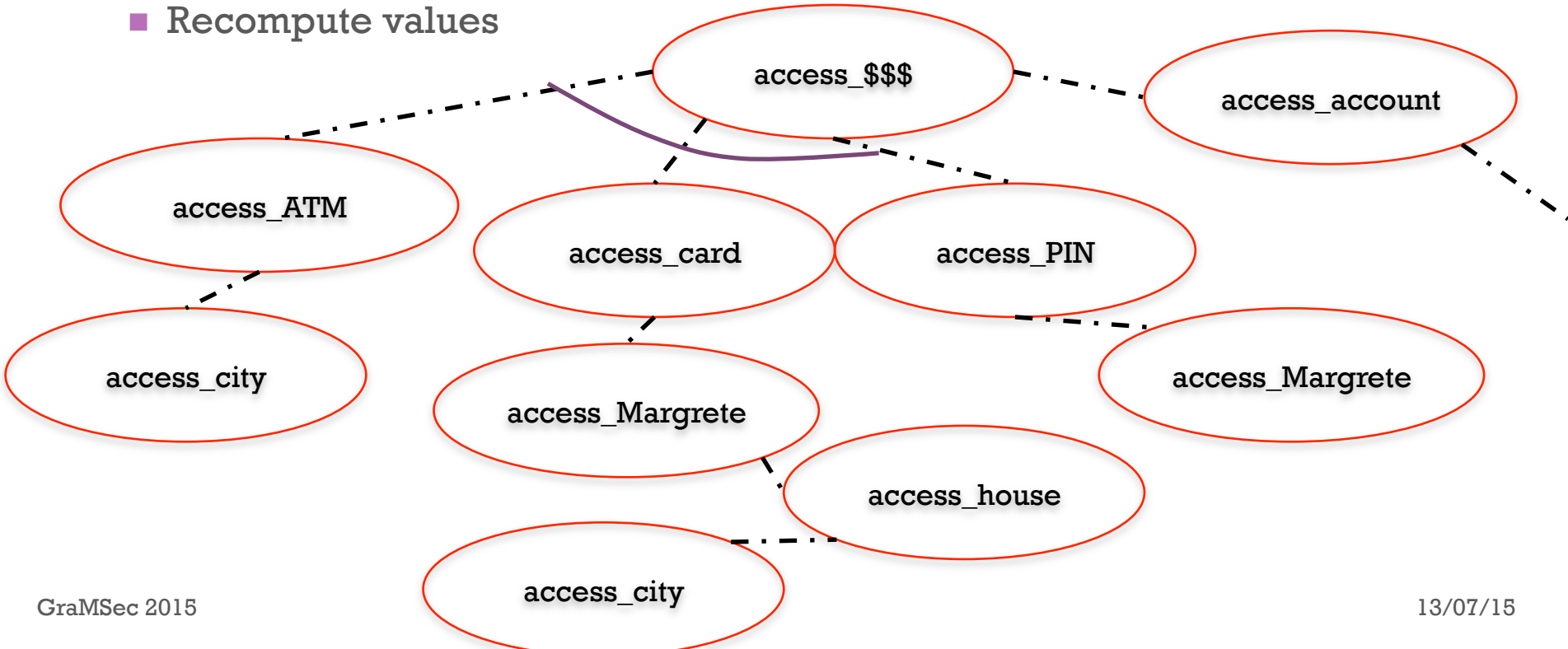
  compute what is accessible and evaluate attack-defence trees using bundle values in the the propositional semantics

- Bootstrapping:
  - For every element n and actor p

  Accessible (n, p) = Reachable (n,p) AND Granted (n,p)

Attack-defence trees synthesis III

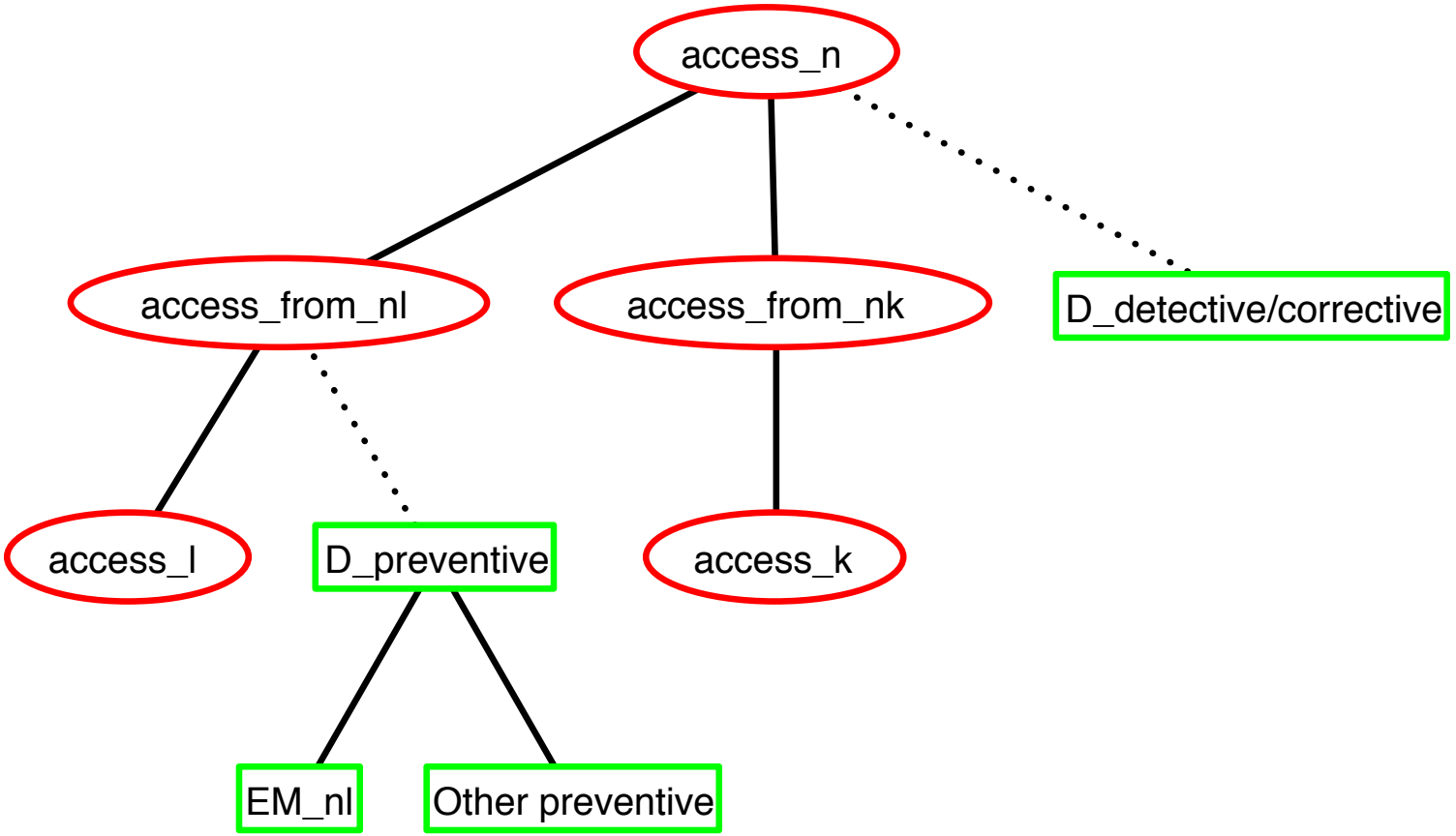
- For a chosen asset t and attacker a
  - Set initial value of each bundle as Accessible (t, a)
  - Synthesize attack-defence trees from individual bundles
  - Expand each bundle only once
  - Recompute values



What about other defences?

- Attack-defence bundles form the initial attack-defence model generated from the socio-technical model
- After the bundles were generated, new controls can be added into individual bundles
  - Consistency is maintained because each single bundle corresponds to access to a single model element
- Placement of new controls depends on their types:
  - Preventive
  - Detective
  - Corrective

New controls: where



How to select new controls

- Proposals for optimal countermeasure selection exist if possible options are already known and evaluated by experts [Roy et al. 2012], [Aslanyan et al. 2015]
- BUT how to assist the experts in selecting new controls consistently from a set of recommended best practices (e.g., NIST 800-53) ?
- Possible considerations:
  - Application domain of controls (model element types)
  - Attributes to be evaluated

Application domains of controls

Entity	Physical space	Digital space
Preventive		
Location	Physical access control	Technical access control, firewall
Actor	Physical access control, Security trainings, Email filter	Technical access control and authentication
Object	Physical access control	Technical access control
Detective		
Location Actor Object	Security cameras, visitor logs	System logs, IDS
Corrective		
Location Actor Object	Insurance, liability limitation, business continuity plan	Insurance, liability limitation, secure state restoring mechanisms, business continuity plan

Attributes

Attribute	Preventive	Detective	Corrective
Risk of detection		✓	
Cost of attack (for attacker)	✓		
Probability of attack success	✓		
Time of attack	✓		
Impact of attack	✓	✓	✓

Challenges ahead

- **Extending the attack-defence model by using an attack-defence library**
  - Knowledge how an attacker can break enforcement mechanisms
  - Knowledge from industry catalogues
- **Socio-technical attacks**
  - Trust policies
  - More complex models with processes
- **Validation**
  - <usefulness> how suitable is the attack-defence model proposed for maintaining defences across system evolution?
  - <scalability> is it possible to generate meaningful attack-defence trees for realistic socio-technical models?
- **Minimal representation and visualization**
  - Attack-defence trees generated will require some restructuring for minimizing the size and excluding redundancies
- **Assisted defence selection**
  - How to guide experts to select optimal countermeasures (to which extent the defences can be generated)?

Conclusions

- Defence generation from socio-technical models is limited by the models themselves
- Attack-defence model consisting of individual attack-defence bundles can help to select and maintain defences across the system lifecycle
- It is easier to generate attacks than defences

Thank you!!!