RWTH AACHEN UNIVERSITY

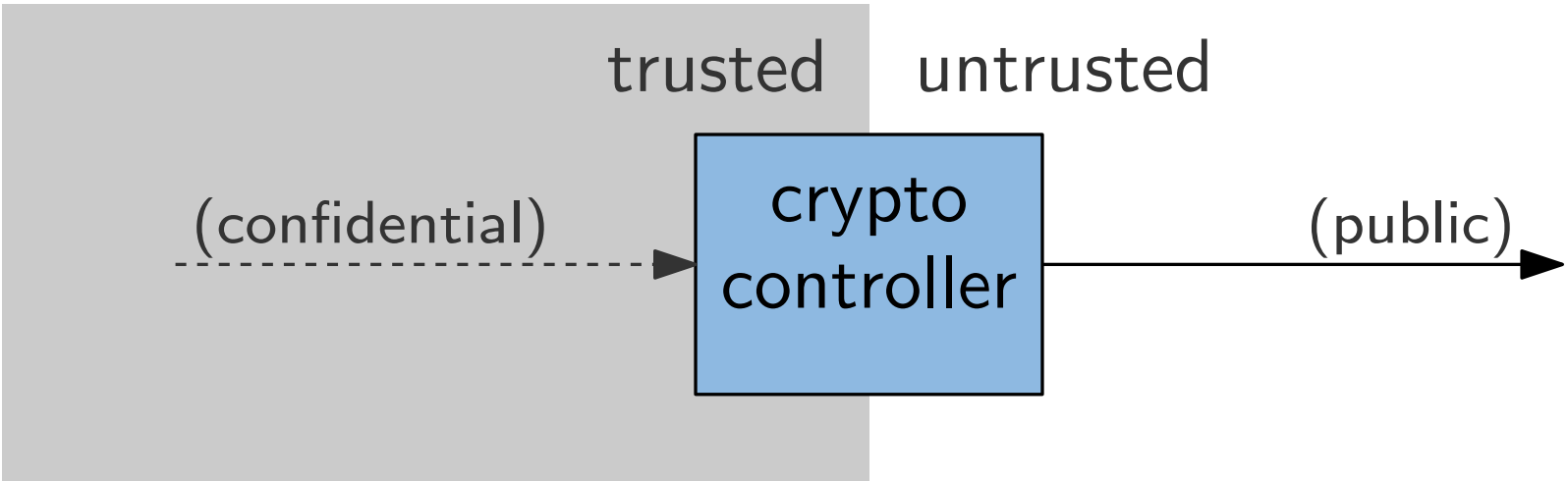
Security Type Checking for MILS-AADL Specifications

Kevin van der Pol, Thomas Noll

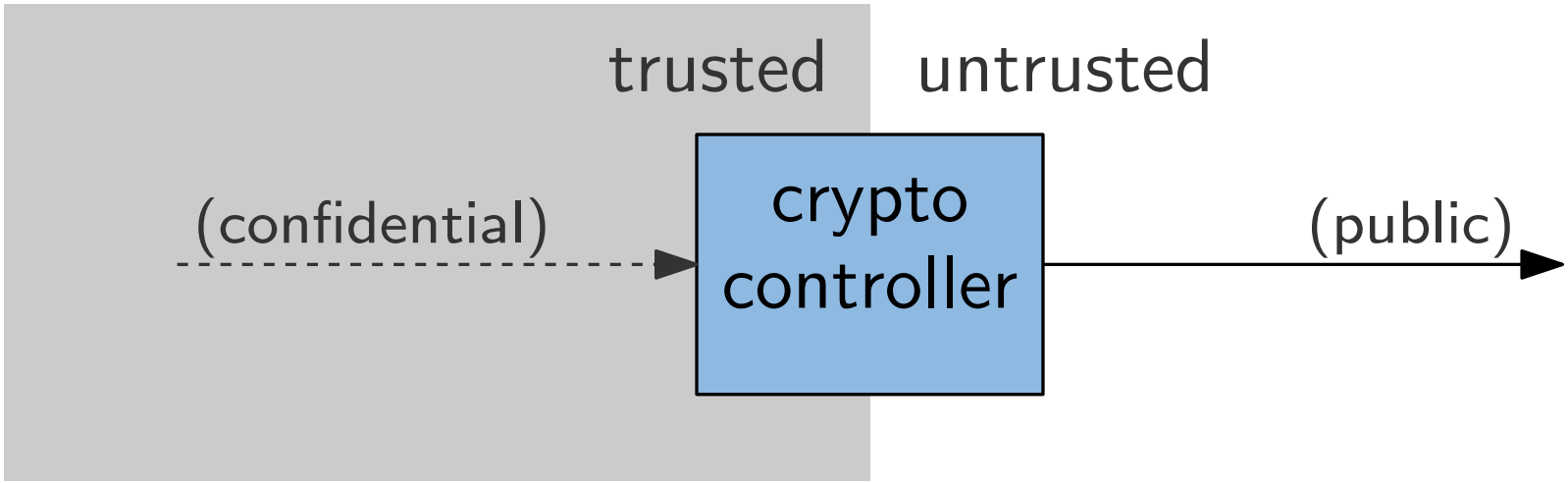
MILS Workshop Amsterdam – January 20, 2015

RWTH AACHEN UNIVERSITY

Motivation



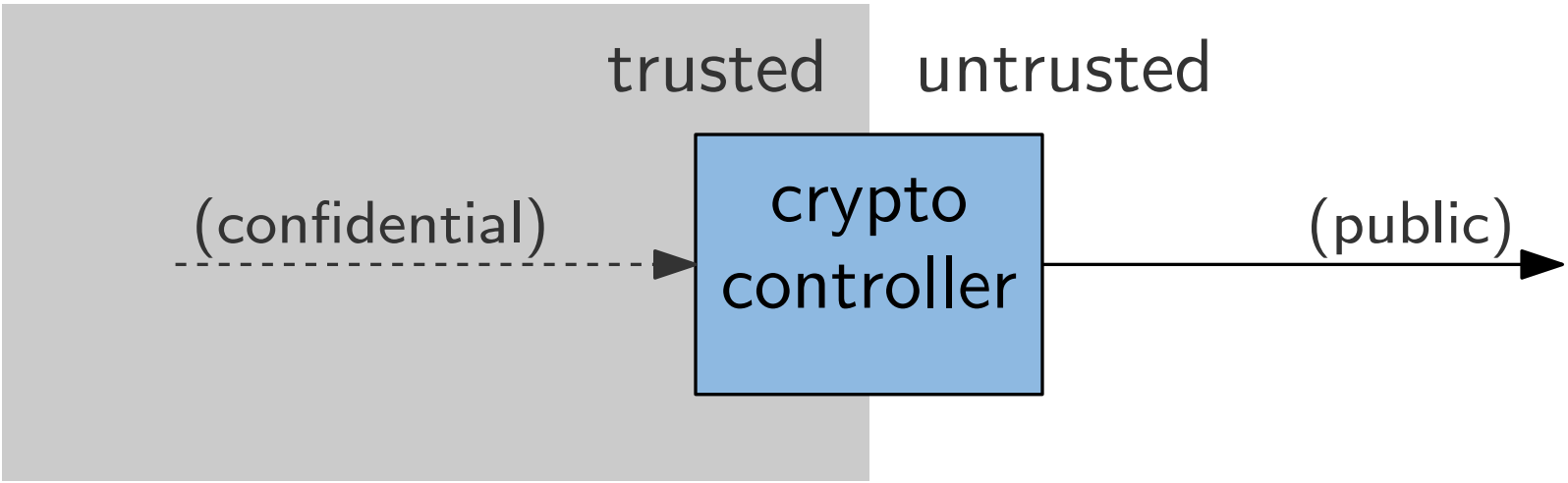
Motivation



- Cryptographic controller [Rus92]

RWTH AACHEN UNIVERSITY

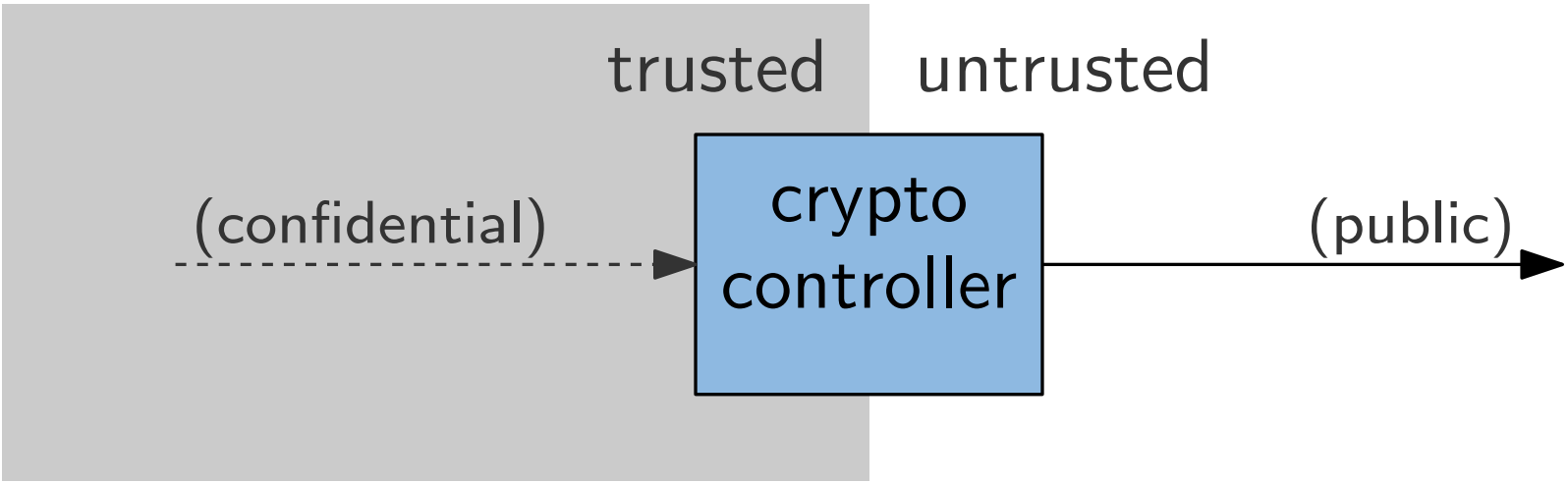
Motivation



- Cryptographic controller [Rus92]
- Placed between trusted system and untrusted network

RWTH AACHEN UNIVERSITY

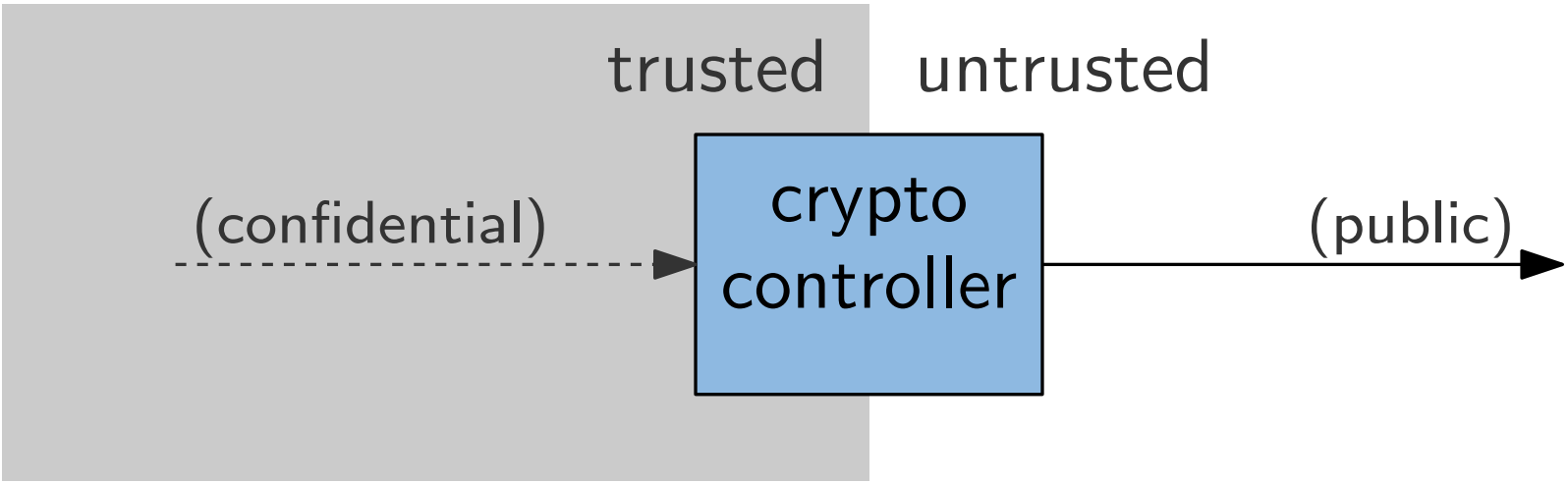
Motivation



- Cryptographic controller [Rus92]
- Placed between trusted system and untrusted network
- Declassifies confidential information

RWTH AACHEN UNIVERSITY

Motivation



- Cryptographic controller [Rus92]
- Placed between trusted system and untrusted network
- Declassifies confidential information
- Goal: verify its security

RWTH AACHEN UNIVERSITY

Table of Contents

RWTH AACHEN UNIVERSITY

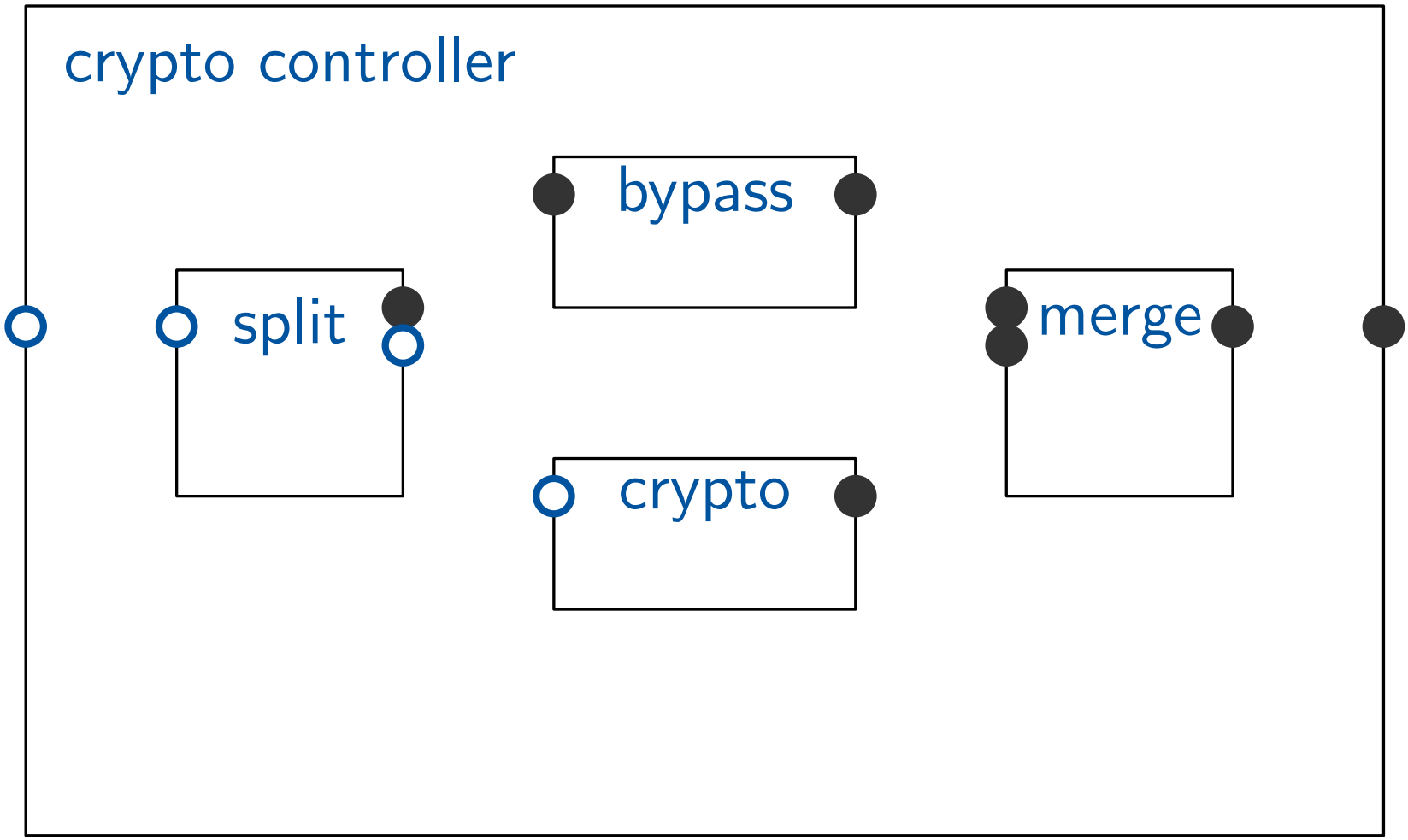
Table of Contents

RWTH AACHEN UNIVERSITY

Crypto controller

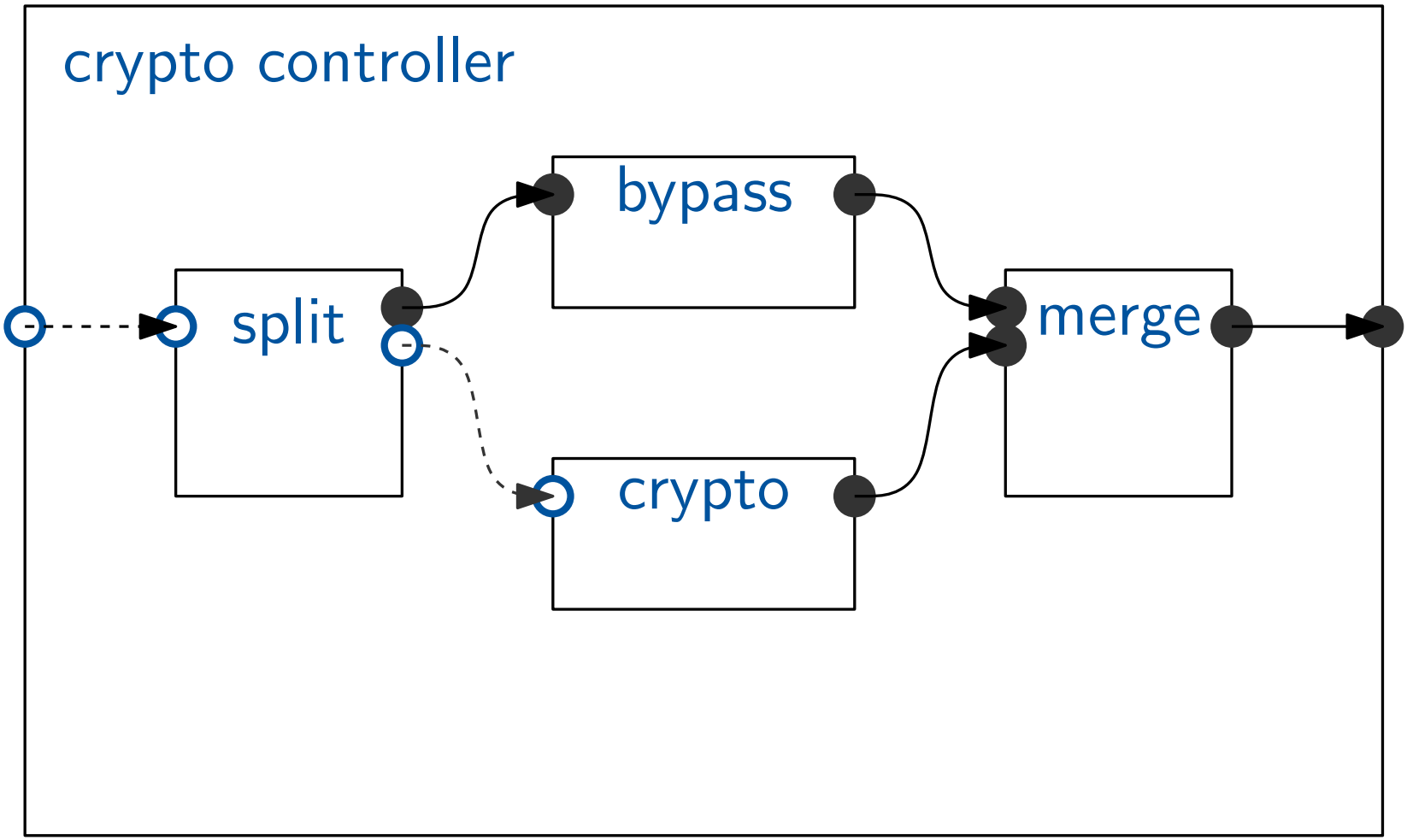
RWTH AACHEN UNIVERSITY

Crypto controller



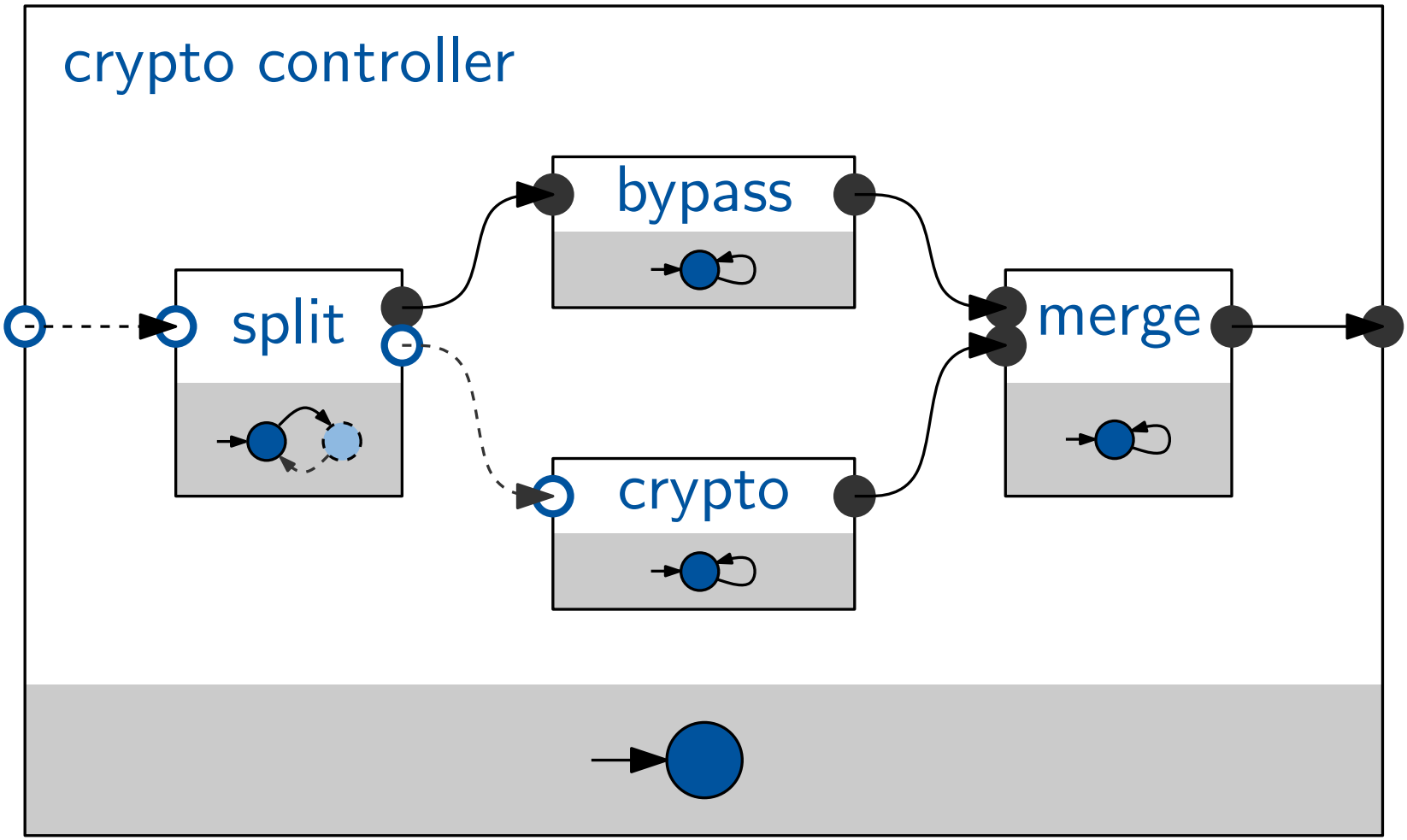
RWTH AACHEN UNIVERSITY

Crypto controller



RWTH AACHEN UNIVERSITY

Crypto controller



RWTH AACHEN UNIVERSITY

Syntax

system cryptocontroller(

)

RWTH AACHEN UNIVERSITY

Syntax

```
system cryptocontroller(
   inframe: in (int L, int H) (0,0)
   outframe: out (int L, enc int H L) (0, encrypt(0, k0))




)
```

RWTH AACHEN UNIVERSITY

Syntax

```
system cryptocontroller(
   inframe: in (int L, int H) (0,0)
   outframe: out (int L, enc int H L) (0, encrypt(0, k0))
   system split(...)
   system bypass(...)
   system merge(...)
   system crypto(
      inpayload: in int H 0
      outpayload: out enc int H L encrypt(0, k0)
      k: key L k0


   )



)
```

RWTH AACHEN UNIVERSITY

Syntax

```
system cryptocontroller(
   inframe: in (int L, int H) (0,0)
   outframe: out (int L, enc int H L) (0, encrypt(0, k0))
   system split(...)
   system bypass(...)
   system merge(...)
   system crypto(
      inpayload: in int H 0
      outpayload: out enc int H L encrypt(0, k0)
      k: key L k0


   )
   connection (split.payload, crypto.inpayload)
   connection (crypto.outpayload, merge.payload)
   ⋮
)
```

RWTH AACHEN UNIVERSITY

Syntax

```
system cryptocontroller(
   inframe: in (int L, int H) (0,0)
   outframe: out (int L, enc int H L) (0, encrypt(0, k0))
   system split(...)
   system bypass(...)
   system merge(...)
   system crypto(
      inpayload: in int H 0
      outpayload: out enc int H L encrypt(0, k0)
      k: key L k0
      m: initial mode L
      m-[then outpayload:=encrypt(inpayload,k)]->m
   )
   connection (split.payload, crypto.inpayload)
   connection (crypto.outpayload, merge.payload)
   ⋮
)
```

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$
System	$S ::= \texttt{system}\ s(S^*\ P^*\ C^*\ V^*\ M^*\ T^*)$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$
System	$S ::= \texttt{system}\ s(S^*\ P^*\ C^*\ V^*\ M^*\ T^*)$
Port	$P ::= p : (\texttt{in} \mid \texttt{out})(\texttt{event}\ \sigma \mid \texttt{data}\ \tau\ e)$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$
System	$S ::= \texttt{system}\ s(S^*\ P^*\ C^*\ V^*\ M^*\ T^*)$
Port	$P ::= p : (\texttt{in} \mid \texttt{out})(\texttt{event}\ \sigma \mid \texttt{data}\ \tau\ e)$
Connection	$C ::= ([s.]p, [s.]p)$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$
System	$S ::= \texttt{system}\ s(S^*\ P^*\ C^*\ V^*\ M^*\ T^*)$
Port	$P ::= p : (\texttt{in} \mid \texttt{out})(\texttt{event}\ \sigma \mid \texttt{data}\ \tau\ e)$
Connection	$C ::= ([s.]p, [s.]p)$
Variable	$V ::= x : \tau\ e$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$
System	$S ::= \texttt{system}\ s(S^*\ P^*\ C^*\ V^*\ M^*\ T^*)$
Port	$P ::= p : (\texttt{in} \mid \texttt{out})(\texttt{event}\ \sigma \mid \texttt{data}\ \tau\ e)$
Connection	$C ::= ([s.]p, [s.]p)$
Variable	$V ::= x : \tau\ e$
Mode	$M ::= m : [\texttt{initial}]\ \texttt{mode}\ \sigma$

RWTH AACHEN UNIVERSITY

Syntax

Case	Grammar
Level	$\sigma ::= \texttt{H} \mid \texttt{L}$
Basic type	$t ::= \texttt{int} \mid \texttt{bool} \mid \texttt{enc}\ \tau$
Security type	$\tau ::= t\ \sigma \mid \texttt{key}\ \sigma$
Expression	$e ::= n \mid x \mid e \oplus e$
System	$S ::= \texttt{system}\ s(S^*\ P^*\ C^*\ V^*\ M^*\ T^*)$
Port	$P ::= p : (\texttt{in} \mid \texttt{out})(\texttt{event}\ \sigma \mid \texttt{data}\ \tau\ e)$
Connection	$C ::= ([s.]p, [s.]p)$
Variable	$V ::= x : \tau\ e$
Mode	$M ::= m : [\texttt{initial}]\ \texttt{mode}\ \sigma$
Transition	$T ::= m\ \texttt{-[}\ [p]\ [\texttt{when}\ e]\ [\texttt{then}\ x := e]\ \texttt{]->}\ m'$

RWTH AACHEN UNIVERSITY

Table of Contents

RWTH AACHEN UNIVERSITY

Non-interference

- Lattice of confidentiality levels

RWTH AACHEN UNIVERSITY

Non-interference

- Lattice of confidentiality levels
- This presentation: high and low ($\mathtt{H} \sqsubseteq \mathtt{L}$)

RWTH AACHEN UNIVERSITY

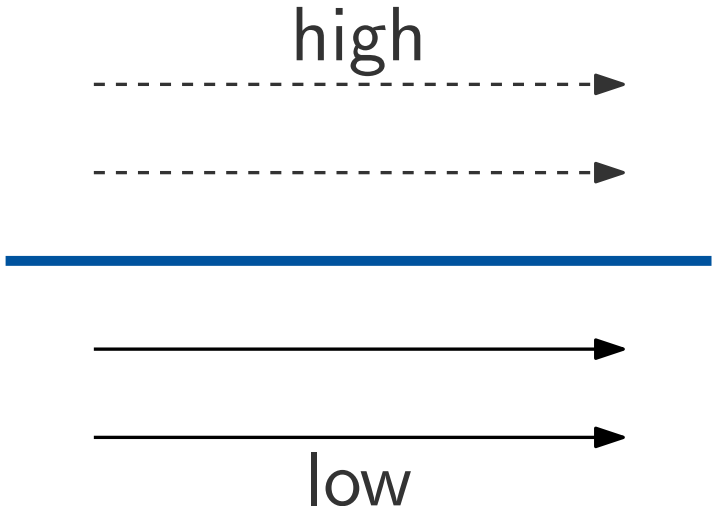
Non-interference

- Lattice of confidentiality levels
- This presentation: high and low ($\mathtt{H} \sqsubseteq \mathtt{L}$)
- Non-interference: low outputs not affected by high inputs

RWTH AACHEN UNIVERSITY

Non-interference

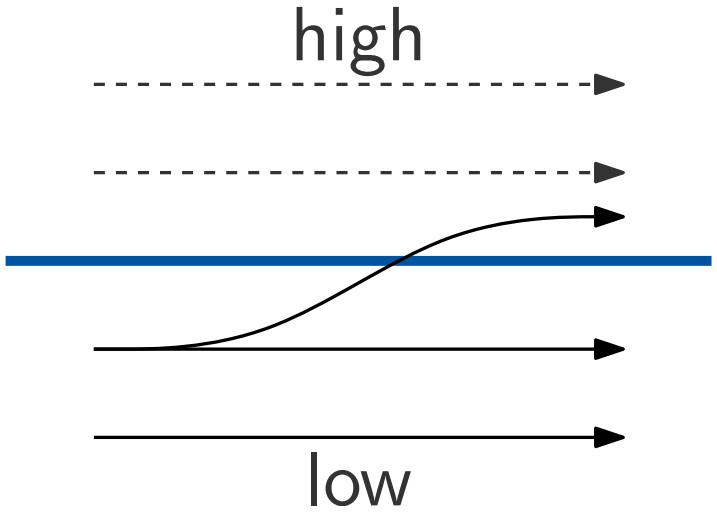
- Lattice of confidentiality levels
- This presentation: high and low ($\mathtt{H} \sqsubseteq \mathtt{L}$)
- Non-interference: low outputs not affected by high inputs



RWTH AACHEN UNIVERSITY

Non-interference

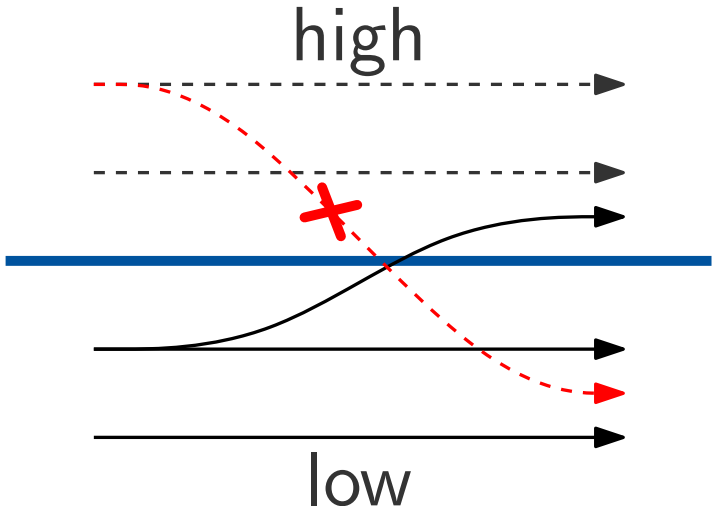
- Lattice of confidentiality levels
- This presentation: high and low ($\mathtt{H} \sqsubseteq \mathtt{L}$)
- Non-interference: low outputs not affected by high inputs



RWTH AACHEN UNIVERSITY

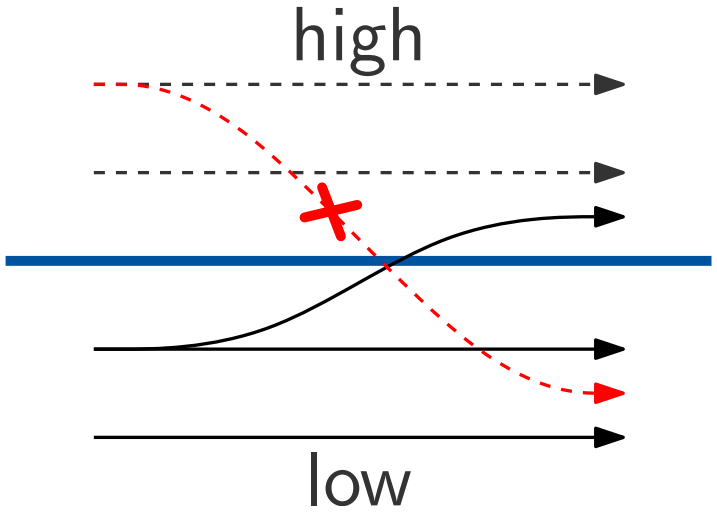
Non-interference

- Lattice of confidentiality levels
- This presentation: high and low ($\mathtt{H} \sqsubseteq \mathtt{L}$)
- Non-interference: low outputs not affected by high inputs



Non-interference

- Lattice of confidentiality levels
- This presentation: high and low ($\mathtt{H} \sqsubseteq \mathtt{L}$)
- Non-interference: low outputs not affected by high inputs



- When varying secret variables and ports, reachable values of public variables and ports are indistinguishable

RWTH AACHEN UNIVERSITY

Unwanted data flows

- Explicit data flow

RWTH AACHEN UNIVERSITY

Unwanted data flows

- Explicit data flow
  - physical connection from high to low ports

RWTH AACHEN UNIVERSITY

Unwanted data flows

- Explicit data flow
  - physical connection from high to low ports
  - low := high

RWTH AACHEN UNIVERSITY

Unwanted data flows

- Explicit data flow
  - physical connection from high to low ports
  - low := high
- Implicit data flow

Unwanted data flows

- Explicit data flow
  - physical connection from high to low ports
  - low := high
- Implicit data flow
  - data leak through control flow

RWTH AACHEN UNIVERSITY

Unwanted data flows

- Explicit data flow
  - physical connection from high to low ports
  - low := high
- Implicit data flow
  - data leak through control flow
  - `if` high `then` low := 1 `else` low := 2 `endif`

Possibilistic non-interference

- Abstract encryption function

RWTH AACHEN UNIVERSITY

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference
  - Public ciphertexts do depend on confidential contents!

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference
  - Public ciphertexts do depend on confidential contents!
- Encryption non-deterministically calculates a ciphertext from a set

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference
  - Public ciphertexts do depend on confidential contents!
- Encryption non-deterministically calculates a ciphertext from a set
- Look at sets of possibilities

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference
  - Public ciphertexts do depend on confidential contents!

- Encryption non-deterministically calculates a ciphertext from a set
- Look at sets of possibilities
- Naive approach: all ciphertexts are indistinguishable

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference
  - Public ciphertexts do depend on confidential contents!

- Encryption non-deterministically calculates a ciphertext from a set
- Look at sets of possibilities
- Naive approach: all ciphertexts are indistinguishable
- Cannot distinguish calculating new ciphertext or re-using same one

Possibilistic non-interference

- Abstract encryption function
- Encryption breaks traditional non-interference
  - Public ciphertexts do depend on confidential contents!

- Encryption non-deterministically calculates a ciphertext from a set
- Look at sets of possibilities
- Naive approach: all ciphertexts are indistinguishable
- Cannot distinguish calculating new ciphertext or re-using same one
  - low1 := encrypt(v,k); low2 := low1

Possibilistic non-interference

- Indistinguishability equivalence $\doteq$ on ciphertexts [AHS08]

Possibilistic non-interference

- Indistinguishability equivalence $\doteq$ on ciphertexts [AHS08]
  - safe usage

Possibilistic non-interference

- Indistinguishability equivalence $\doteq$ on ciphertexts [AHS08]
  - safe usage
  - prevent implicit flow

RWTH AACHEN UNIVERSITY

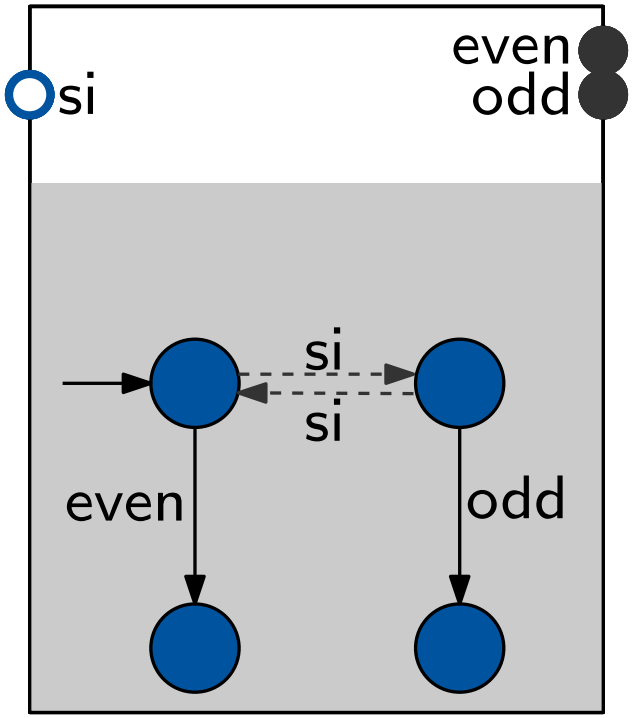
Possibilistic non-interference

- Indistinguishability equivalence $\doteq$ on ciphertexts [AHS08]
  - safe usage
  - prevent implicit flow
- Realistic for common encryption classes (IND-CPA, INT-PTXT) [Lau08]

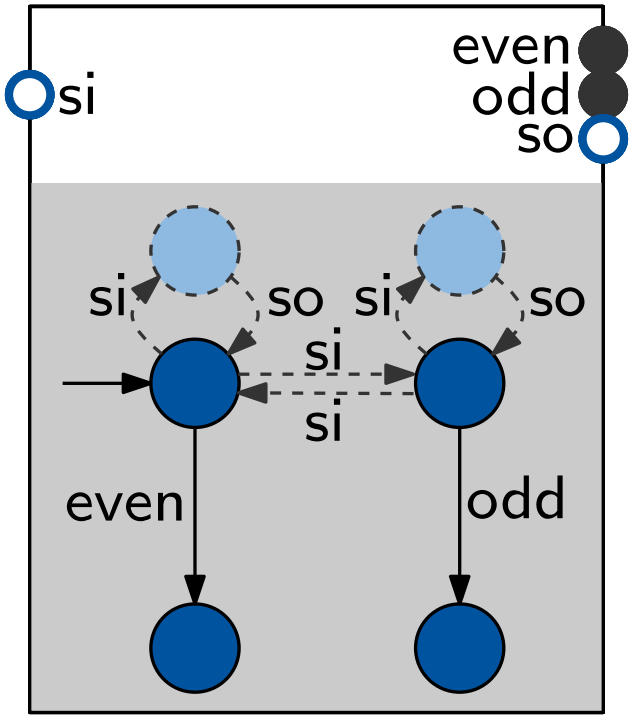
Possibilistic non-interference

- Indistinguishability equivalence $\doteq$ on ciphertexts [AHS08]
  - safe usage
  - prevent implicit flow
- Realistic for common encryption classes (IND-CPA, INT-PTXT) [Lau08]
- Possibilistic non-interference

Non-interference non-compositionality

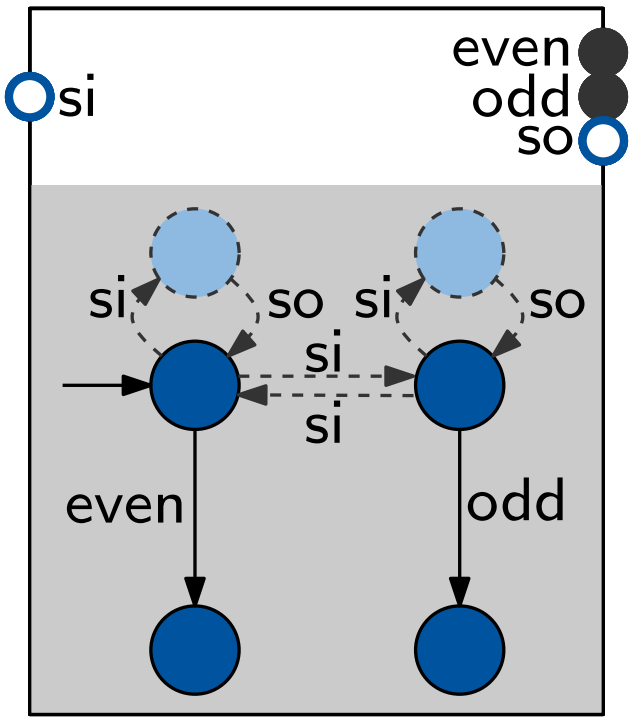


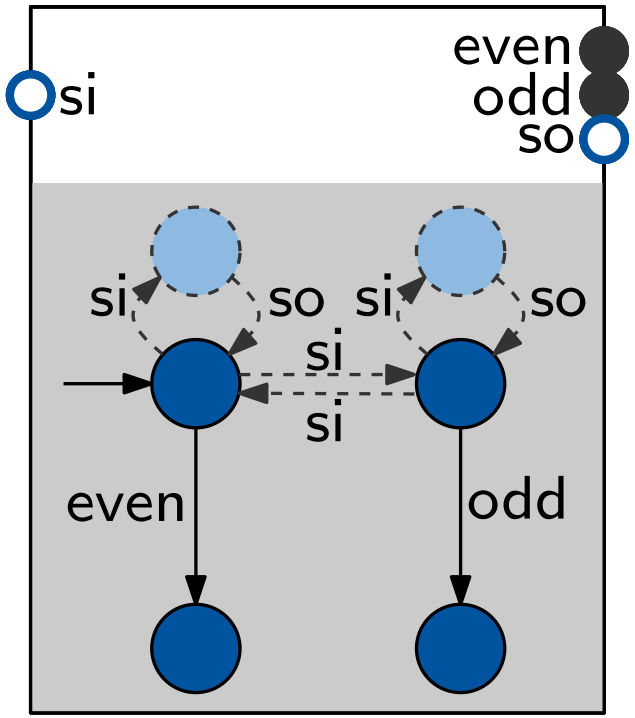
Non-interference non-compositionality



RWTH AACHEN UNIVERSITY

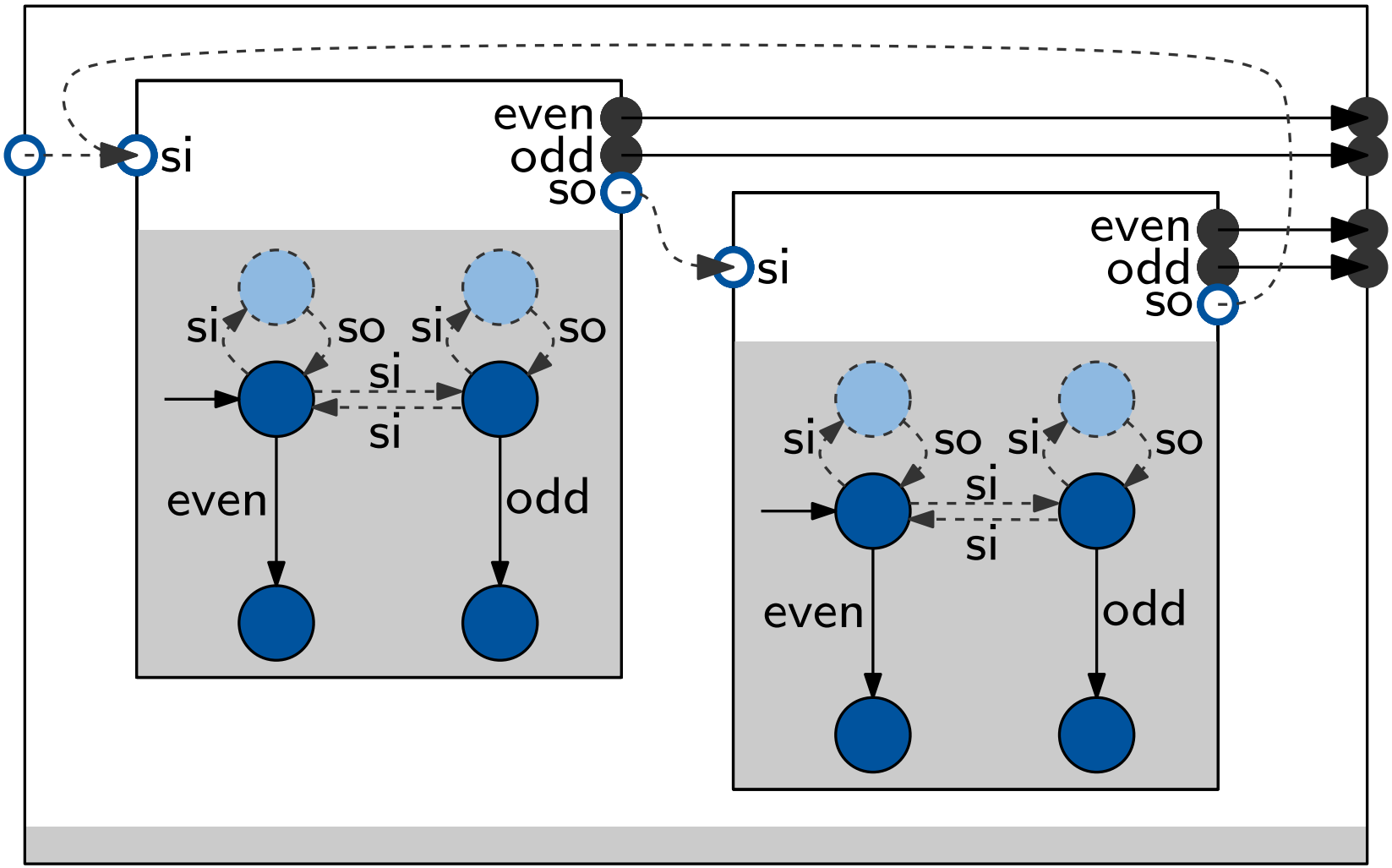
Non-interference non-compositionality



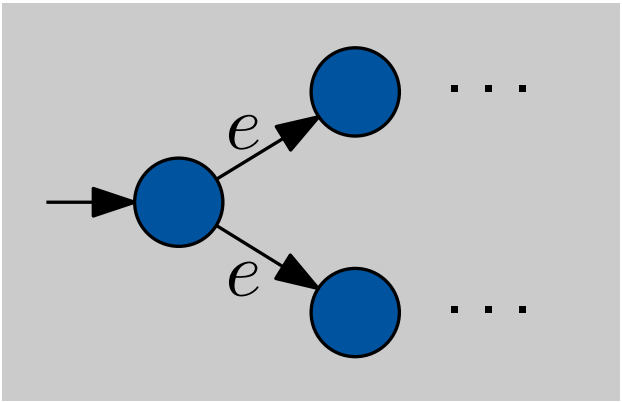


RWTH AACHEN UNIVERSITY

Non-interference non-compositionality



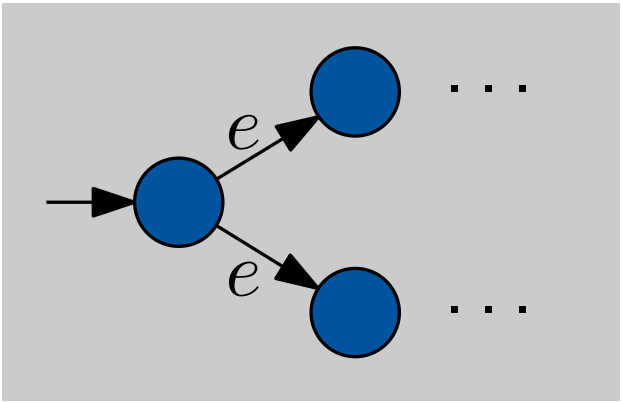
Restrictions



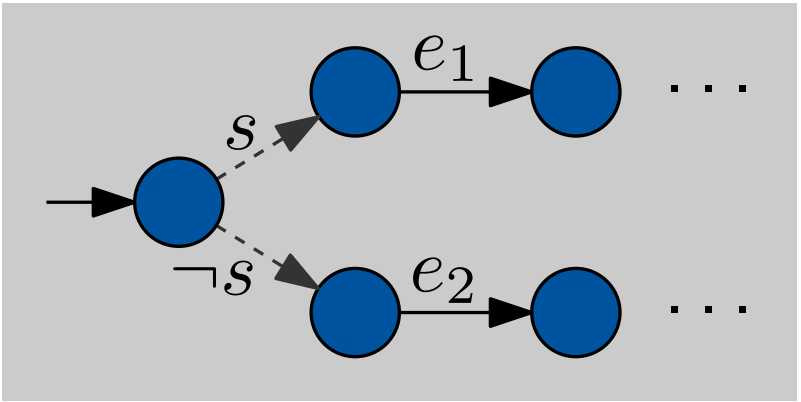
non-determinism

RWTH AACHEN UNIVERSITY

Restrictions



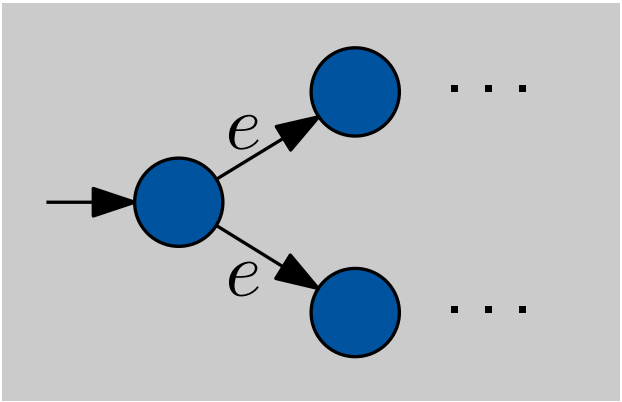
non-determinism



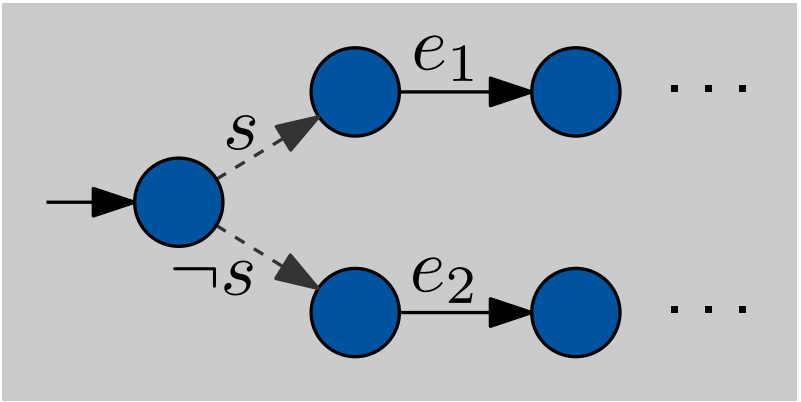
branching on secrets

RWTH AACHEN UNIVERSITY

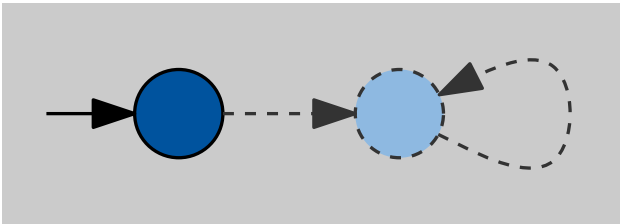
Restrictions



non-determinism



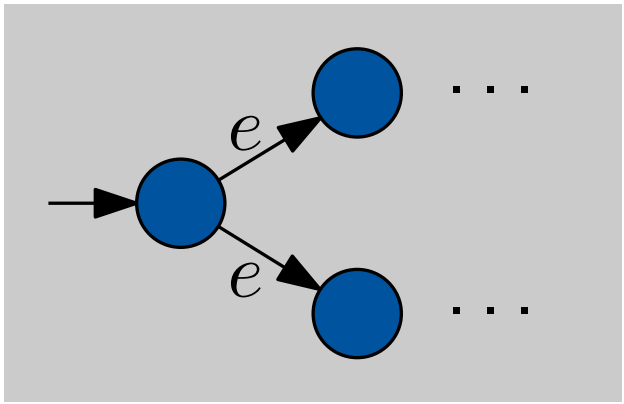
branching on secrets



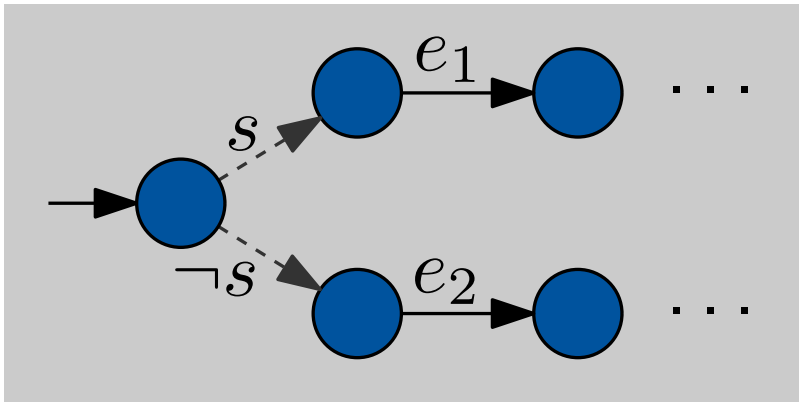
secret-Zeno

RWTH AACHEN UNIVERSITY

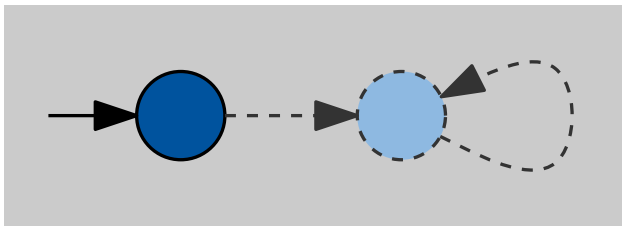
Restrictions



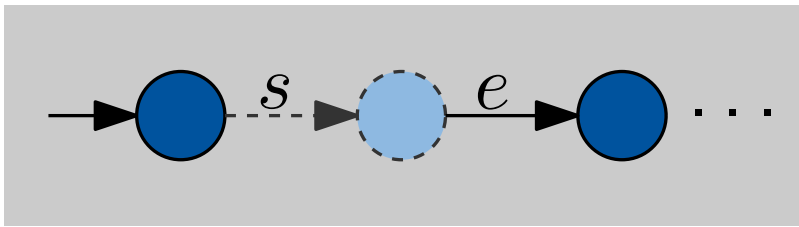
non-determinism



branching on secrets



secret-Zeno



public output from secret region

RWTH AACHEN UNIVERSITY

Table of Contents

Type checking

- T: local variables and data ports $\rightarrow$ declared type

Type checking

- T: local variables and data ports $\rightarrow$ declared type
- Modes and event ports get a confidentiality level

RWTH AACHEN UNIVERSITY

Type checking

- T: local variables and data ports $\rightarrow$ declared type
- Modes and event ports get a confidentiality level
- Type rules in context of T

RWTH AACHEN UNIVERSITY

Type checking

- T: local variables and data ports $\rightarrow$ declared type
- Modes and event ports get a confidentiality level
- Type rules in context of T
- Goal: type each subsystem, connection and transition

RWTH AACHEN UNIVERSITY

Type checking

- T: local variables and data ports $\rightarrow$ declared type
- Modes and event ports get a confidentiality level
- Type rules in context of T
- Goal: type each subsystem, connection and transition

Type checking

- T: local variables and data ports $\rightarrow$ declared type
- Modes and event ports get a confidentiality level
- Type rules in context of T
- Goal: type each subsystem, connection and transition

Theorem

If the system is typable, it is non-interfering

Types of expressions

Case	Type rule
Basic types	$\dfrac{}{T \vdash n : \texttt{int L}} \qquad \dfrac{}{T \vdash b : \texttt{bool L}}$
Variable	$\dfrac{T(x) = \tau}{T \vdash x : \tau}$
Operator	$\dfrac{T \vdash e_1 : t_1\ \sigma_1 \quad T \vdash e_2 : t_2\ \sigma_2 \quad \oplus : t_1 \times t_2 \to t}{T \vdash e_1 \oplus e_2 : t\ (\sigma_1 \sqcup \sigma_2)}$
Encryption	$\dfrac{T \vdash e_1 : \tau \quad T \vdash e_2 : \texttt{key L}}{T \vdash \texttt{encrypt}(e_1, e_2) : \texttt{enc}\ \tau\ \texttt{L}}$
Decryption	$\dfrac{T \vdash e_1 : \texttt{enc}\ \tau\ \sigma \quad T \vdash e_2 : \texttt{key H}}{T \vdash \texttt{decrypt}(e_1, e_2) : \tau^{\sigma}}$

Types of expressions

```
system crypto(
    inpayload: int H 0
    outpayload: out enc int H L encrypt(0, k0)
    k: key L k0
    m: initial mode L
    m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Type of `encrypt(inpayload,k)`?

$$\frac{\dfrac{T(\texttt{inpayload}) = \texttt{int H}}{T \vdash \texttt{inpayload} : \texttt{int H}} \quad \dfrac{T(\texttt{k}) = \texttt{key L}}{T \vdash \texttt{k} : \texttt{key L}}}{T \vdash \texttt{encrypt(inpayload,k)} : \texttt{enc int H L}}$$

Subtyping

- If $\tau_e <: \tau_x$, assignment $x := e$ is valid
  - Traditional type safety
  - Confidentiality restriction

Subtyping

- If $\tau_e <: \tau_x$, assignment $x := e$ is valid
  - Traditional type safety
  - Confidentiality restriction

Case	Subtype rule
Basic types	$\dfrac{\sigma \sqsubseteq \sigma'}{\texttt{int}\ \sigma <: \texttt{int}\ \sigma'}$ $\dfrac{\sigma \sqsubseteq \sigma'}{\texttt{bool}\ \sigma <: \texttt{bool}\ \sigma'}$ $\dfrac{}{\texttt{key}\ \sigma <: \texttt{key}\ \sigma}$

Subtyping

- If $\tau_e <: \tau_x$, assignment $x := e$ is valid
  - Traditional type safety
  - Confidentiality restriction

Case	Subtype rule
Basic types	$\dfrac{\sigma \sqsubseteq \sigma'}{\texttt{int}\ \sigma <: \texttt{int}\ \sigma'}$ $\dfrac{\sigma \sqsubseteq \sigma'}{\texttt{bool}\ \sigma <: \texttt{bool}\ \sigma'}$ $\dfrac{}{\texttt{key}\ \sigma <: \texttt{key}\ \sigma}$
Encryption	$\dfrac{\tau <: \tau' \quad \sigma \sqsubseteq \sigma'}{\texttt{enc}\ \tau\ \sigma <: \texttt{enc}\ \tau'\ \sigma'}$

Subtyping

```
system crypto(
    inpayload: int H 0
    outpayload: out enc int H L encrypt(0, k0)
    k: key L k0
    m: initial mode L
    m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is `outpayload:=encrypt(inpayload,k)` valid?

Subtyping

```
system crypto(
   inpayload: int H 0
   outpayload: out enc int H L encrypt(0, k0)
   k: key L k0
   m: initial mode L
   m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is `outpayload:=encrypt(inpayload,k)` valid?
  (let τ abbreviate `enc int H L`)

Subtyping

```
system crypto(
   inpayload: int H 0
   outpayload: out enc int H L encrypt(0, k0)
   k: key L k0
   m: initial mode L
   m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is `outpayload:=encrypt(inpayload,k)` valid?
  (let τ abbreviate `enc int H L`)

$$\frac{\dfrac{T(\texttt{outpayload}) = \tau}{T \vdash \texttt{outpayload} : \tau} \quad \dfrac{\textit{(Before)}}{T \vdash \texttt{encrypt(inpayload,k)} : \tau} \quad \tau <: \tau}{T \vdash \texttt{outpayload:=encrypt(inpayload,k)} : \tau}$$

Type of transitions

Case	Type rule
Transition	$$\frac{\begin{array}{lll} T \vdash p : \tau_p & & \\ T \vdash g : \tau_g & \tau_x <: \tau_e & lvl(\tau_x) \sqsubseteq \sigma \\ T \vdash x : \tau_x & lvl(m) \sqsubseteq lvl(\tau_p) & \sigma \sqsubseteq lvl(\tau_p) \\ T \vdash e : \tau_e & lvl(m) \sqsubseteq lvl(\tau_x) & \sigma \sqsubseteq lvl(\tau_g) \end{array}}{T \vdash m \texttt{-[}\, p \ \texttt{when}\ g\ \texttt{then}\ x := e \,\texttt{]->}\, m' : \sigma}$$

Type of transitions

Case	Type rule
Transition	$\dfrac{\begin{array}{lll} T \vdash p : \tau_p & & \\ T \vdash g : \tau_g & \tau_x <: \tau_e & lvl(\tau_x) \sqsubseteq \sigma \\ T \vdash x : \tau_x & lvl(m) \sqsubseteq lvl(\tau_p) & \sigma \sqsubseteq lvl(\tau_p) \\ T \vdash e : \tau_e & lvl(m) \sqsubseteq lvl(\tau_x) & \sigma \sqsubseteq lvl(\tau_g) \end{array}}{T \vdash m \texttt{-[}\, p \ \texttt{when}\ g\ \texttt{then}\ x := e \,\texttt{]->}\, m' : \sigma}$

Type of transitions

Case	Type rule
Transition	$$\frac{\begin{array}{lll} T \vdash p : \tau_p & & \\ T \vdash g : \tau_g & \tau_x <: \tau_e & lvl(\tau_x) \sqsubseteq \sigma \\ T \vdash x : \tau_x & lvl(m) \sqsubseteq lvl(\tau_p) & \sigma \sqsubseteq lvl(\tau_p) \\ T \vdash e : \tau_e & lvl(m) \sqsubseteq lvl(\tau_x) & \sigma \sqsubseteq lvl(\tau_g) \end{array}}{T \vdash m \texttt{-[}\, p \;\texttt{when}\; g \;\texttt{then}\; x := e \,\texttt{]->}\, m' : \sigma}$$

- Effect is well typed

Type of transitions

Case	Type rule
Transition	$\dfrac{\begin{array}{lll} T \vdash p : \tau_p & & \\ T \vdash g : \tau_g & \tau_x <: \tau_e & lvl(\tau_x) \sqsubseteq \sigma \\ T \vdash x : \tau_x & lvl(m) \sqsubseteq lvl(\tau_p) & \sigma \sqsubseteq lvl(\tau_p) \\ T \vdash e : \tau_e & lvl(m) \sqsubseteq lvl(\tau_x) & \sigma \sqsubseteq lvl(\tau_g) \end{array}}{T \vdash m \texttt{-[} p \texttt{ when } g \texttt{ then } x := e \texttt{]->} m' : \sigma}$

- Effect is well typed
- No public outputs from secret region (mode)

Type of transitions

Case	Type rule
Transition	$\dfrac{\begin{array}{lll} T \vdash p : \tau_p & & \\ T \vdash g : \tau_g & \tau_x <: \tau_e & lvl(\tau_x) \sqsubseteq \sigma \\ T \vdash x : \tau_x & lvl(m) \sqsubseteq lvl(\tau_p) & \sigma \sqsubseteq lvl(\tau_p) \\ T \vdash e : \tau_e & lvl(m) \sqsubseteq lvl(\tau_x) & \sigma \sqsubseteq lvl(\tau_g) \end{array}}{T \vdash m \texttt{-[} p \texttt{ when } g \texttt{ then } x := e \texttt{]->} m' : \sigma}$

- Effect is well typed
- No public outputs from secret region (mode)
- Guard or event high $\rightarrow$ transition high

Type of transitions

Case	Type rule
Transition	$\dfrac{\begin{array}{lll} T \vdash p : \tau_p & & \\ T \vdash g : \tau_g & \tau_x <: \tau_e & lvl(\tau_x) \sqsubseteq \sigma \\ T \vdash x : \tau_x & lvl(m) \sqsubseteq lvl(\tau_p) & \sigma \sqsubseteq lvl(\tau_p) \\ T \vdash e : \tau_e & lvl(m) \sqsubseteq lvl(\tau_x) & \sigma \sqsubseteq lvl(\tau_g) \end{array}}{T \vdash m \texttt{-[}\, p \;\texttt{when}\; g \;\texttt{then}\; x := e \,\texttt{]->}\, m' : \sigma}$

- Effect is well typed
- No public outputs from secret region (mode)
- Guard or event high $\rightarrow$ transition high
- No public outputs from secret region (transition)

RWTH AACHEN UNIVERSITY

Example

```
system crypto(
    inpayload: int H 0
    outpayload: out enc int H L encrypt(0, k0)
    k: key L k0
    m: initial mode L
    m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?

RWTH AACHEN UNIVERSITY

Example

```
system crypto(
    inpayload: int H 0
    outpayload: out enc int H L encrypt(0, k0)
    k: key L k0
    m: initial mode L
    m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?
  - Effect is well typed: ✓

Example

```
system crypto(
    inpayload: int H 0
    outpayload: out enc int H L encrypt(0, k0)
    k: key L k0
    m: initial mode L
    m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?
  - Effect is well typed: ✓
  - No public outputs from secret region (mode): ✓

Example

```
system crypto(
   inpayload: int H 0
   outpayload: out enc int H L encrypt(0, k0)
   k: key L k0
   m: initial mode L
   m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?
  - Effect is well typed: ✓
  - No public outputs from secret region (mode): ✓
  - Guard or event high $\rightarrow$ transition high: ✓

Example

```
system crypto(
   inpayload: int H 0
   outpayload: out enc int H L encrypt(0, k0)
   k: key L k0
   m: initial mode L
   m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?
  - Effect is well typed: ✓
  - No public outputs from secret region (mode): ✓
  - Guard or event high $\rightarrow$ transition high: ✓
  - No public outputs from secret region (transition): ✓

Example

```
system crypto(
    inpayload: int H 0
    outpayload: out enc int H L encrypt(0, k0)
    k: key L k0
    m: initial mode L
    m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?
  - Effect is well typed: ✓
  - No public outputs from secret region (mode): ✓
  - Guard or event high $\rightarrow$ transition high: ✓
  - No public outputs from secret region (transition): ✓
- All subsystems and connections trivially well typed

Example

```
system crypto(
   inpayload: int H 0
   outpayload: out enc int H L encrypt(0, k0)
   k: key L k0
   m: initial mode L
   m-[then outpayload:=encrypt(inpayload,k)]->m
)
```

- Is the transition well typed?
  - Effect is well typed: ✓
  - No public outputs from secret region (mode): ✓
  - Guard or event high $\rightarrow$ transition high: ✓
  - No public outputs from secret region (transition): ✓
- All subsystems and connections trivially well typed
- `crypto` system is non-interfering □

RWTH AACHEN UNIVERSITY

Table of Contents

RWTH AACHEN UNIVERSITY

Conclusion

- AADL variant with cryptographic primitives

RWTH AACHEN UNIVERSITY

Conclusion

- AADL variant with cryptographic primitives
- Automatic verification for possibilistic non-interference

RWTH AACHEN UNIVERSITY

Conclusion

- AADL variant with cryptographic primitives
- Automatic verification for possibilistic non-interference
- Future work: soundness proof, implementation, type inference

RWTH AACHEN UNIVERSITY

Bibliography

Aslan Askarov, Daniel Hedin, and Andrei Sabelfeld, *Cryptographically-masked flows*, Theor. Comput. Sci. **402** (2008), no. 2-3, 82–101.

Peeter Laud, *On the computational soundness of cryptographically masked flows*, Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008, 2008, pp. 337–348.

John Rushby, *Noninterference, transitivity, and channel-control security policies*, Tech. Report SRI-CSL-92-2, Computer Science Laboratory, SRI International, Menlo Park, CA, December 1992.