Universal Composition with Responsive Environments

Jan Camenisch[1], **Robert R. Enderlein**[1], **Stephan Krenn**[2], **Ralf Küsters**[3], **Daniel Rausch**[3]

[1] IBM Research Zurich - Switzerland
[2] AIT - Austria
[3] University of Trier - Germany

Simulation-Based Security

Definition of simulatability (basic idea):

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

Simulation-Based Security

Definition of simulatability (basic idea):

$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$

ideal protocol/functionality
e.g. ideal key exchange

real protocol
e.g. IKE

Simulation-Based Security

Definition of simulatability (basic idea):

realizes

$$\mathcal{P} \leq \mathcal{F}$$

ideal protocol/functionality
e.g. ideal key exchange

real protocol
e.g. IKE

Simulation-Based Security

Definition of simulatability (basic idea):

realizes

$\mathcal{P} \leq \mathcal{F}$ iff

$\mathcal{P}$ $\mathcal{F}$

ideal protocol/functionality
e.g. ideal key exchange

real protocol
e.g. IKE

Simulation-Based Security

Definition of simulatability (basic idea):

realizes

$\mathcal{P} \leq \mathcal{F}$ iff

$\forall$  $\longleftrightarrow$ $\mathcal{P}$ $\mathcal{F}$

ideal protocol/functionality
e.g. ideal key exchange

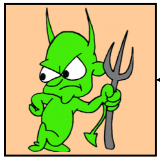
real protocol
e.g. IKE

Simulation-Based Security

Definition of simulatability (basic idea):

realizes

$$\mathcal{P} \leq \mathcal{F} \quad \text{iff}$$

$\forall$  $\longleftrightarrow$ $\mathcal{P}$ $\quad\exists$  $\longleftrightarrow$ $\mathcal{F}$

ideal protocol/functionality
e.g. ideal key exchange

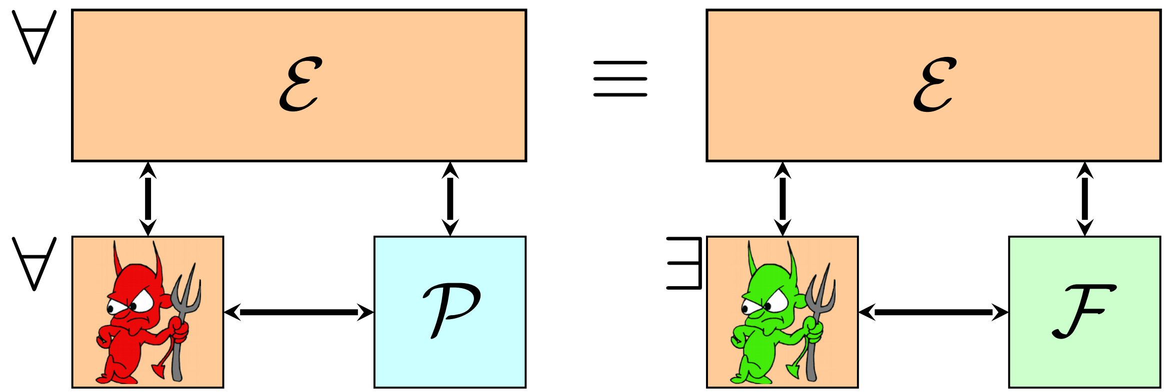
real protocol
e.g. IKE

Simulation-Based Security

Definition of simulatability (basic idea):

realizes

$\mathcal{P} \leq \mathcal{F}$ iff



ideal protocol/functionality
e.g. ideal key exchange

real protocol
e.g. IKE

Compositional Protocol Analysis

Compositional Protocol Analysis

Assume:

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

Compositional Protocol Analysis

Assume:

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g. ideal key exchange

Compositional Protocol Analysis

Assume:

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g. ideal key exchange

Prove:

Compositional Protocol Analysis

Assume:

e.g. ideal key exchange

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g., some real-world protocol
SSL/TLS, SSH, ...

Prove:

$$\boxed{\mathcal{Q}}$$

Compositional Protocol Analysis

e.g. ideal key exchange

Assume:

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g., some real-world protocol
SSL/TLS, SSH, ...

Prove:

$$\boxed{\mathcal{Q}} \leq \boxed{\mathcal{F}'}$$

Compositional Protocol Analysis

e.g. ideal key exchange

Assume:

$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$

e.g., some real-world protocol SSL/TLS, SSH, ...

e.g. ideal secure channel

Prove:

$\boxed{\mathcal{Q}} \leq \boxed{\mathcal{F}'}$

Compositional Protocol Analysis

Assume:

$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$

e.g. ideal key exchange

e.g., some real-world protocol SSL/TLS, SSH, ...

e.g. ideal secure channel

Prove:

$\boxed{\mathcal{Q}} \leq \boxed{\mathcal{F}'}$

Compositional Protocol Analysis

e.g. ideal key exchange

Assume:

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g., some real-world protocol SSL/TLS, SSH, ...

e.g. ideal secure channel

Prove:

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow\\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

Compositional Protocol Analysis

Assume:

$\mathcal{P} \leq \mathcal{F}$

e.g. ideal key exchange

Prove:

e.g., some real-world protocol SSL/TLS, SSH, ...

e.g. ideal secure channel

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

Compositional Protocol Analysis

Assume:

e.g. ideal key exchange

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g., some real-world protocol SSL/TLS, SSH, ...

e.g. ideal secure channel

Prove:

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

Composition Theorem

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{P}}\end{array} \leq \begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

Compositional Protocol Analysis

Assume:

e.g. ideal key exchange

$$\boxed{\mathcal{P}} \leq \boxed{\mathcal{F}}$$

e.g., some real-world protocol SSL/TLS, SSH, ...

e.g. ideal secure channel

Prove:

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

Composition Theorem

$$\begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{P}}\end{array} \leq \begin{array}{c}\boxed{\mathcal{Q}}\\ \updownarrow \\ \boxed{\mathcal{F}}\end{array} \leq \boxed{\mathcal{F}'}$$

can now be used in more complex protocols

Models for Simulation-Based Security

- UC model [Canetti 2001]
- IITM model [Küsters 2006]
- GNUC model [Hofheinz, Shoup 2011]
- ...

What is the problem?

What is the problem?

* Urgent Requests

What is the problem?

* Urgent Requests

* Non-Responsiveness Problem

What is the problem?

* Urgent Requests

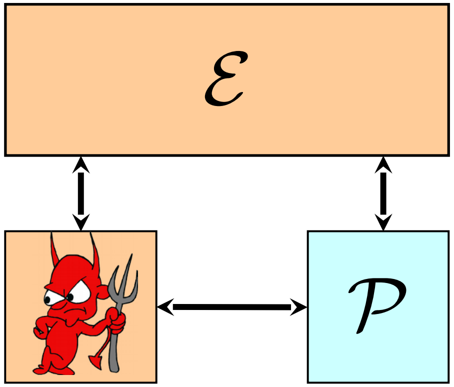
* Non-Responsiveness Problem

Our solution:

Responsive Environments

Urgent Requests

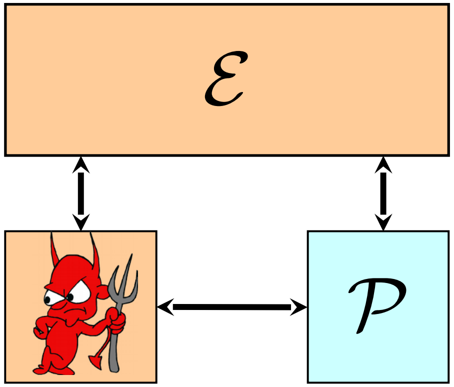
Protocols often have to exchange modeling related meta information with adversary:



Urgent Requests

Protocols often have to exchange modeling related meta information with adversary:

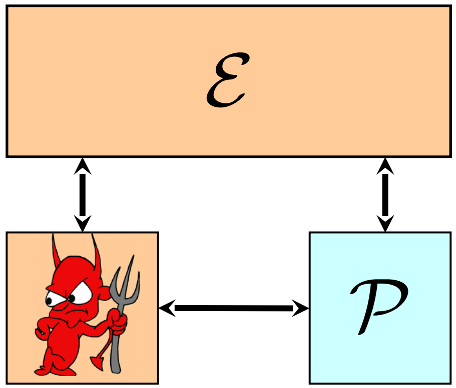
- Ask for corruption status



Urgent Requests

Protocols often have to exchange modeling related meta information with adversary:

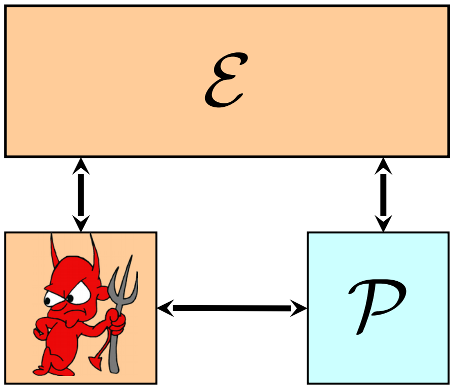
- Ask for corruption status
- Ask for cryptographic material (keys, algorithms,...)



Urgent Requests

Protocols often have to exchange modeling related meta information with adversary:

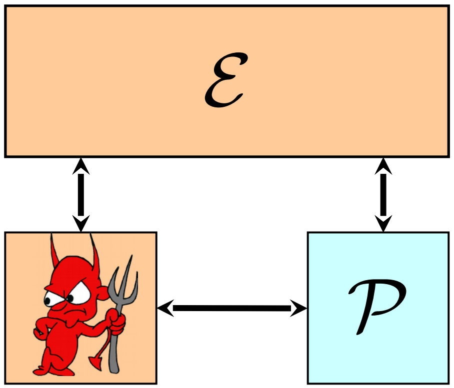
- Ask for corruption status
- Ask for cryptographic material (keys, algorithms,...)
- Leak information



Urgent Requests

Protocols often have to exchange modeling related meta information with adversary:

- Ask for corruption status
- Ask for cryptographic material (keys, algorithms,...)
- Leak information
- Signaling information ("new instance created")

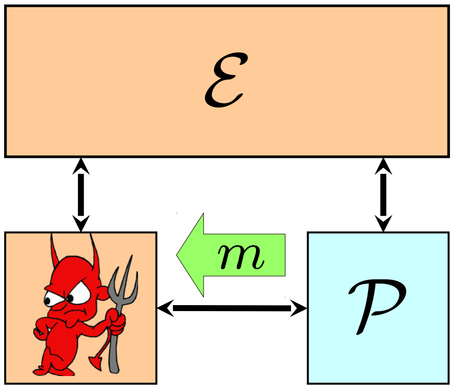


Urgent Requests

Protocols often have to exchange modeling related meta information with adversary:

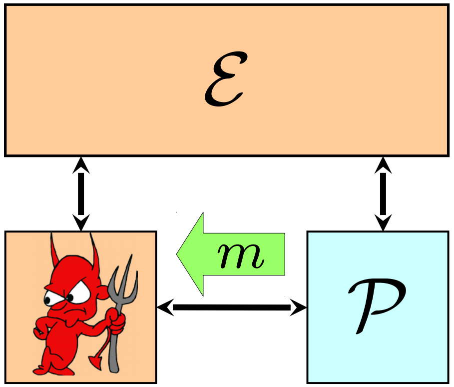
- Ask for corruption status
- Ask for cryptographic material (keys, algorithms,...)
- Leak information
- Signaling information ("new instance created")

$\Rightarrow$ Send a message m (urgent request)



Non-Responsiveness Problem

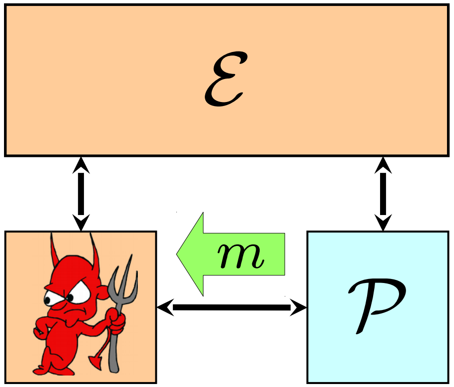
Urgent requests do not model real network traffic



Non-Responsiveness Problem

Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

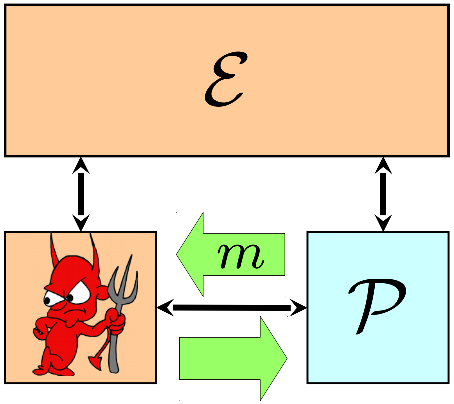


Non-Responsiveness Problem

Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately



Non-Responsiveness Problem

Urgent requests do not model real network traffic

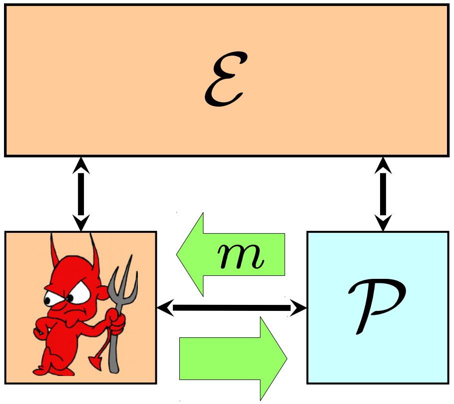
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way



Non-Responsiveness Problem

Urgent requests do not model real network traffic

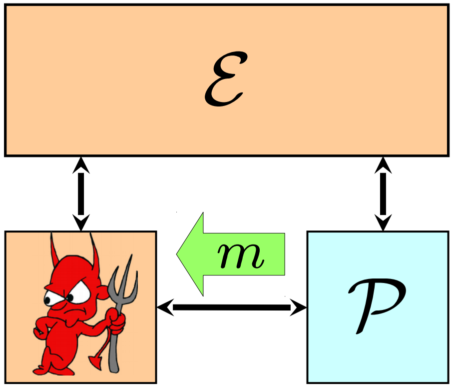
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way



Non-Responsiveness Problem

Urgent requests do not model real network traffic

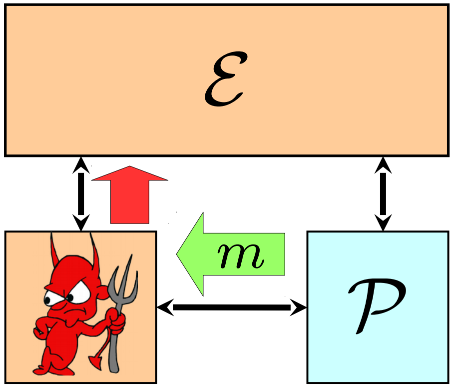
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way



Non-Responsiveness Problem

Urgent requests do not model real network traffic

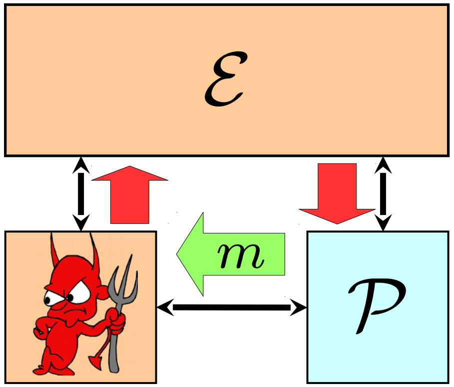
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way



Non-Responsiveness Problem

Urgent requests do not model real network traffic

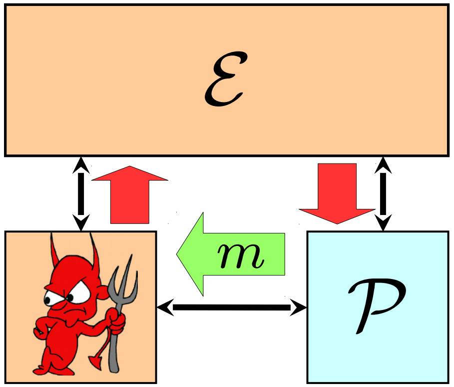
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

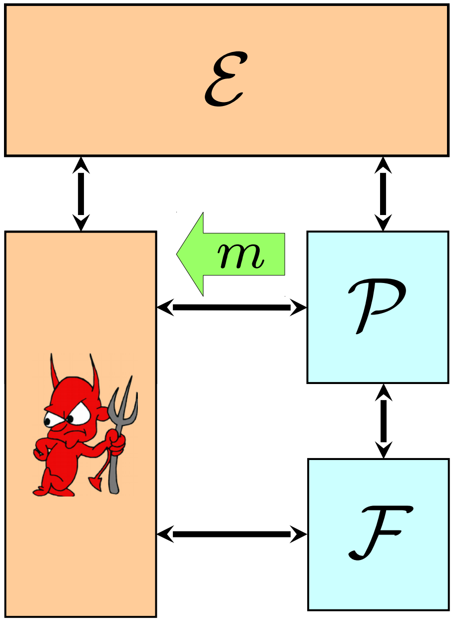
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

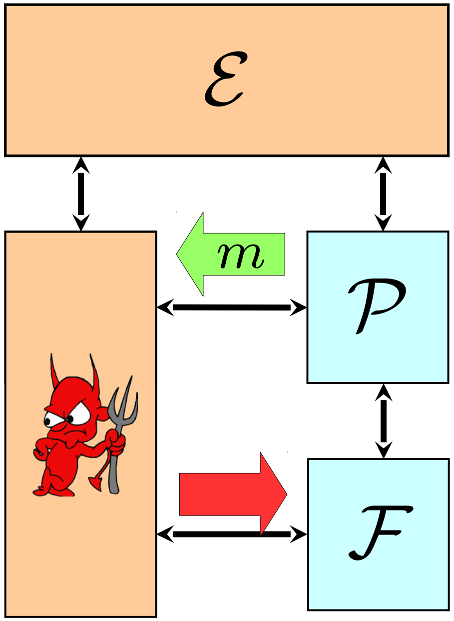
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

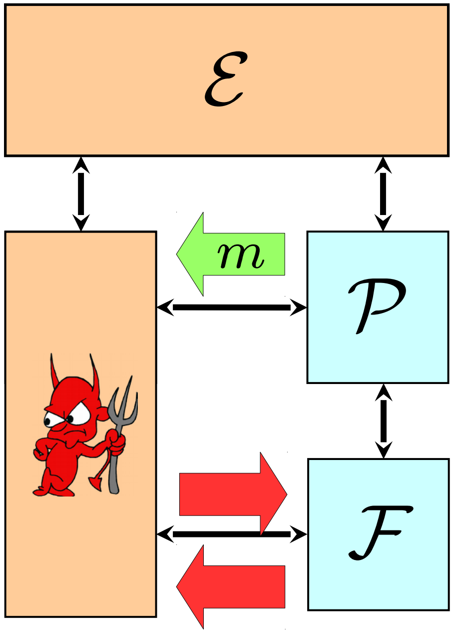
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

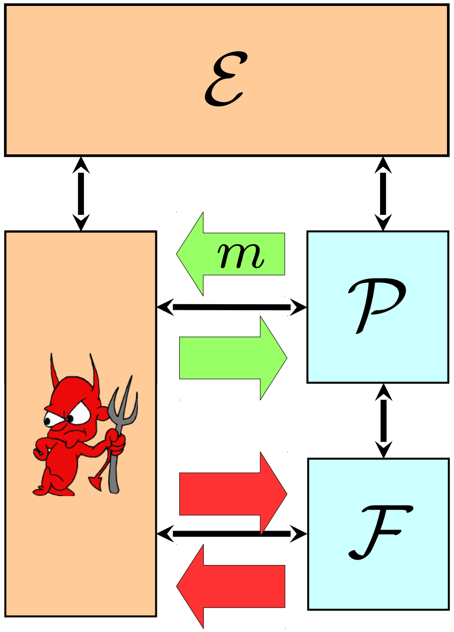
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

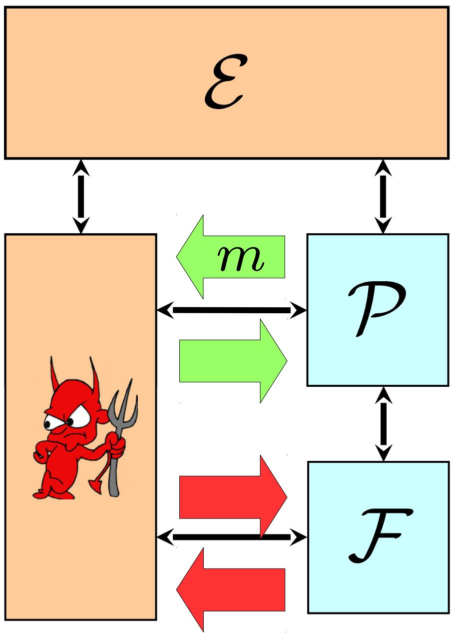
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol
- Block parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

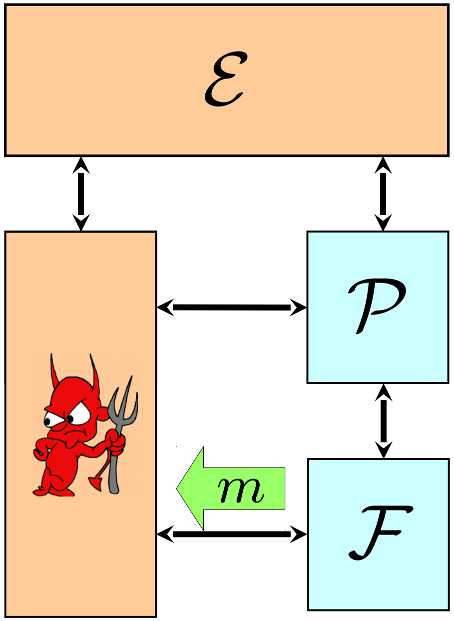
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol
- Block parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

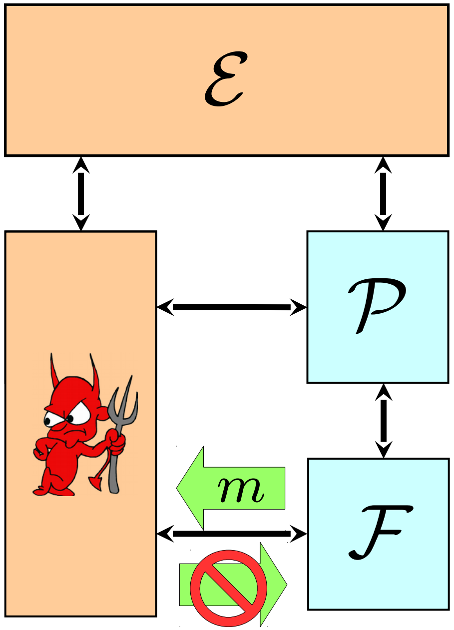
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol
- Block parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

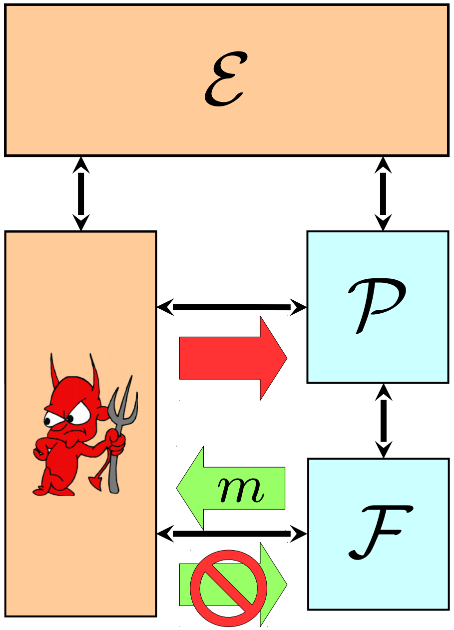
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol
- Block parts of the protocol



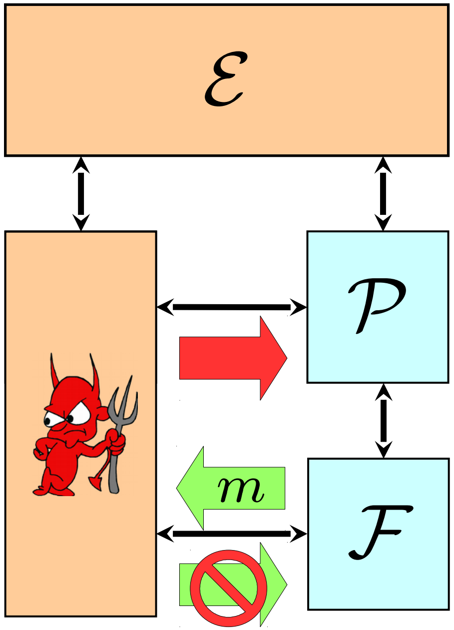
Non-Responsiveness Problem

Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:



Non-Responsiveness Problem

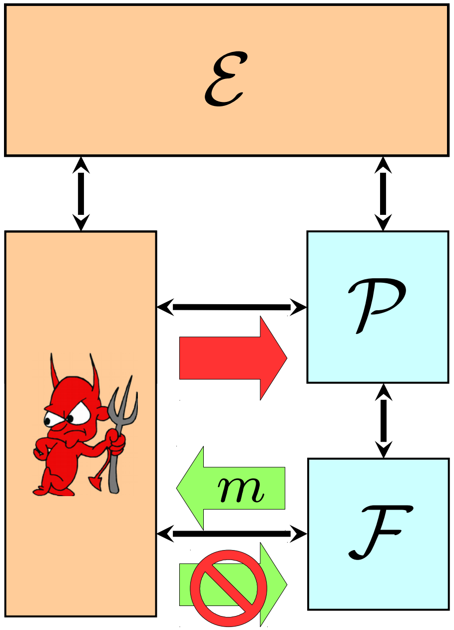
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult



Non-Responsiveness Problem

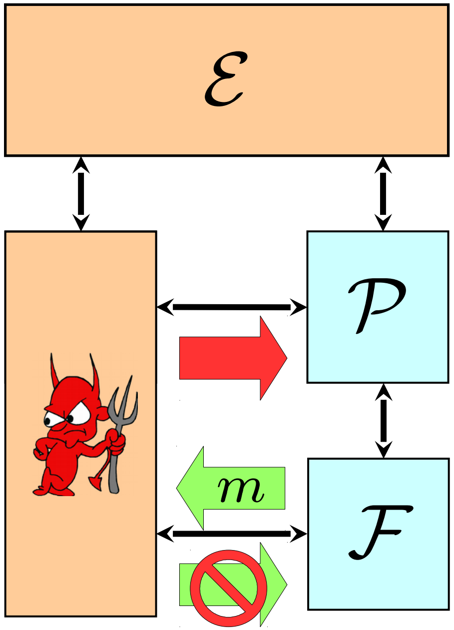
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult
- Not always possible



Non-Responsiveness Problem

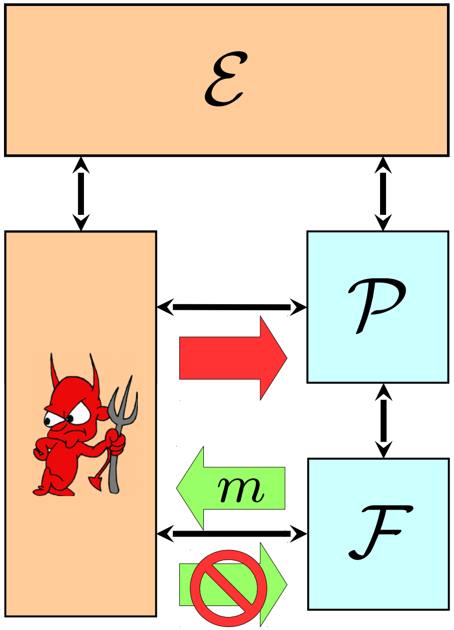
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult
- Not always possible
- Complex specifications and proofs



Non-Responsiveness Problem

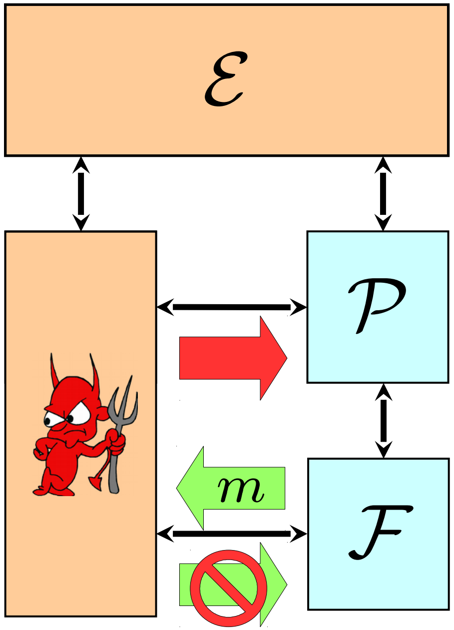
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult
- Not always possible
- Complex specifications and proofs
- Often ignored in the literature



Non-Responsiveness Problem

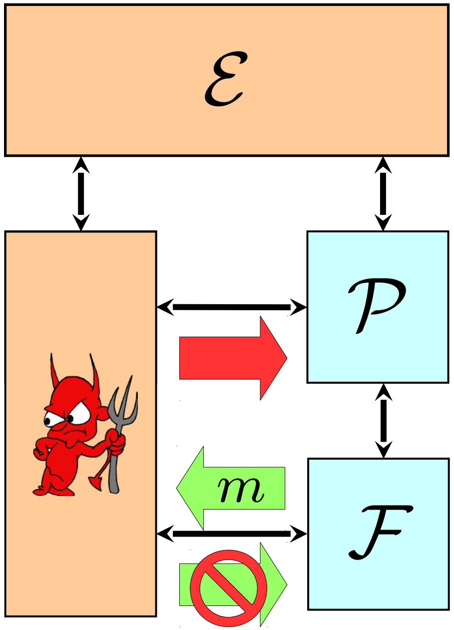
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult
- Not always possible
- Complex specifications and proofs
- Often ignored in the literature
  - * Underspecified protocols



Non-Responsiveness Problem

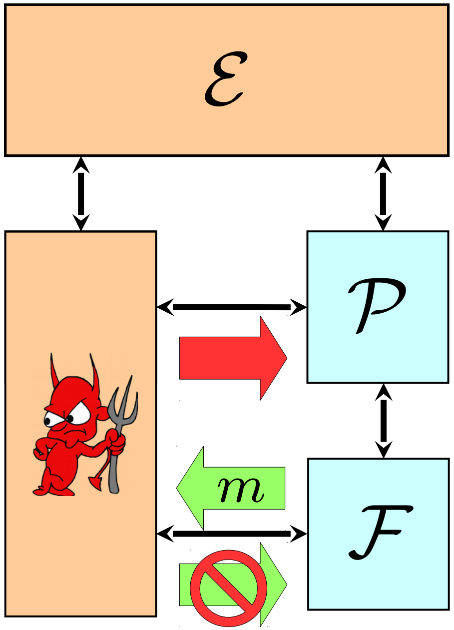
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult
- Not always possible
- Complex specifications and proofs
- Often ignored in the literature
  - * Underspecified protocols
  - * Flawed proofs



Non-Responsiveness Problem

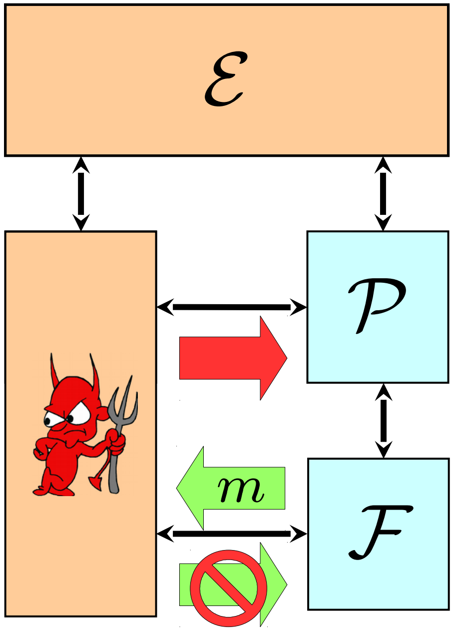
Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Protocol designers have to deal with unintended adversarial behavior:

- Difficult
- Not always possible
- Complex specifications and proofs
- Often ignored in the literature
  - * Underspecified protocols
  - * Flawed proofs
  - * Hard to reuse functionalities



Examples from the literature

$\mathcal{F}_{\text{NIKE}}$ from [Freire, Hesse, Hofheinz, 2014]

Upon input $(\texttt{init}, P_i, P_j)$ from P_i [...] consider two cases:

- Corrupted session mode: if there exists $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, **send $(\texttt{init}, P_i, P_j)$ to the adversary. After receiving $(\{P_i, P_j\}, K_{i,j})$ from the adversary**, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: [...]

Return (P_i, P_j, key) to P_i.

Examples from the literature

$\mathcal{F}_{\mathsf{NIKE}}$ from [Freire, Hesse, Hofheinz, 2014]

Upon input $(\texttt{init}, P_i, P_j)$ from P_i [...] consider two cases:

- Corrupted session mode: if there exists $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, **send $(\texttt{init}, P_i, P_j)$ to the adversary. After receiving $(\{P_i, P_j\}, K_{i,j})$ from the adversary**, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: [...]

Return (P_i, P_j, key) to P_i.

Lack of expressivity:

Functionality meant to model *non-interactive* key exchange, but is actually interactive

Examples from the literature

$\mathcal{F}_{\text{sok}}$ from [Chase, Lysyanskaya, 2006]

Upon receiving a value $(\texttt{Setup}, sid)$ from any party P, verify that $sid = (M_L, sid')$ for some sid'. If not, then ignore the request. Else, if this is the first time that $(\texttt{Setup}, sid)$ was received, **hand $(\texttt{Setup}, sid)$ to the adversary; upon receiving** $(\texttt{Algorithms}, sid, \text{Verify}, \text{Sign}, \text{Simsign}, \text{Extract})$ **from the adversary**, store these algorithms. Output the stored $(\text{Algorithms}, sid, \text{Sign}, \text{Verify})$ to P.

Examples from the literature

$\mathcal{F}_{\text{sok}}$ from [Chase, Lysyanskaya, 2006]

Upon receiving a value $(\texttt{Setup}, sid)$ from any party P, verify that $sid = (M_L, sid')$ for some sid'. If not, then ignore the request. Else, if this is the first time that $(\texttt{Setup}, sid)$ was received, **hand $(\texttt{Setup}, sid)$ to the adversary; upon receiving $(\texttt{Algorithms}, sid, \text{Verify}, \text{Sign}, \text{Simsign}, \text{Extract})$ from the adversary**, store these algorithms. Output the stored $(\text{Algorithms}, sid, \text{Sign}, \text{Verify})$ to P.

Problems in proofs:

Functionality might not receive algorithms,
which is problematic for realizations based on $\mathcal{F}_{\text{sok}}$

Examples from the literature

$\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Upon receiving a value $(\texttt{Verify}, sid, m, \sigma)$ from some party S', **hand $(\texttt{Verify}, sid, m, \sigma)$ to the adversary. Upon receiving $(\texttt{Verified}, sid, m, \phi)$ from the adversary**, do:
[...]
Output $(\texttt{Verified}, sid, m, f)$ to S'.

Examples from the literature

$\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Upon receiving a value $(\texttt{Verify}, sid, m, \sigma)$ from some party S', **hand $(\texttt{Verify}, sid, m, \sigma)$ to the adversary. Upon receiving $(\texttt{Verified}, sid, m, \phi)$ from the adversary**, do:
[...]
Output $(\texttt{Verified}, sid, m, f)$ to S'.

Unintended state changes and behavior:

Adversary can corrupt signer of a signature during verification
$\Rightarrow$ Possible to accept invalid signatures

Examples from the literature

Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.

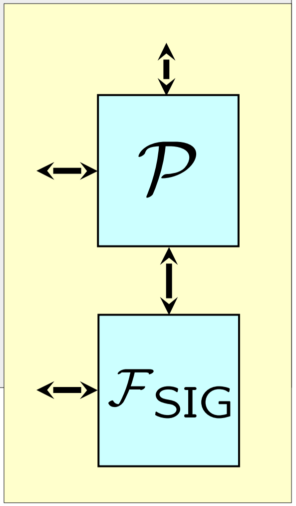
Examples from the literature

Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Examples from the literature

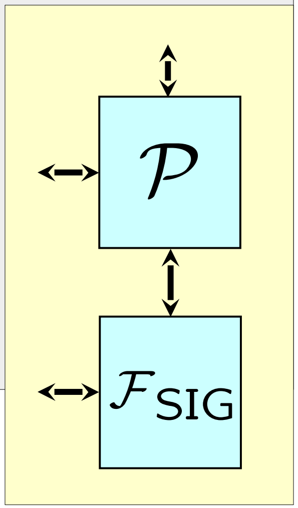
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

subroutine using urgent requests

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Examples from the literature

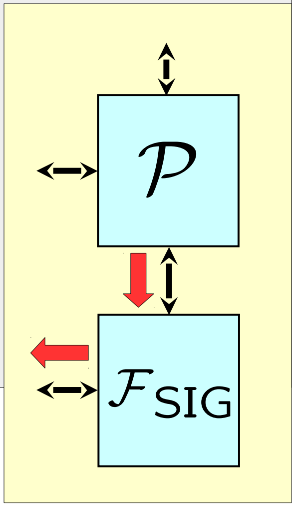
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

subroutine using urgent requests

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Examples from the literature

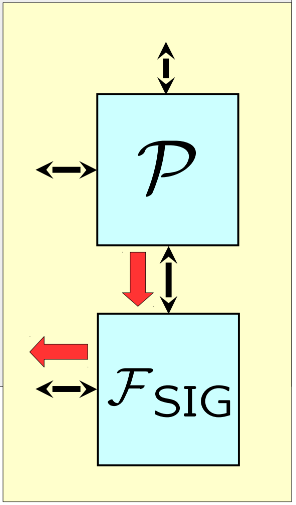
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

subroutine using urgent requests

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Problem propagates to higher level protocols:

Adversary is activated when calling a subroutine which models a local task.

The behavior of $\mathcal{P}$ in this case is undefined.

Examples from the literature

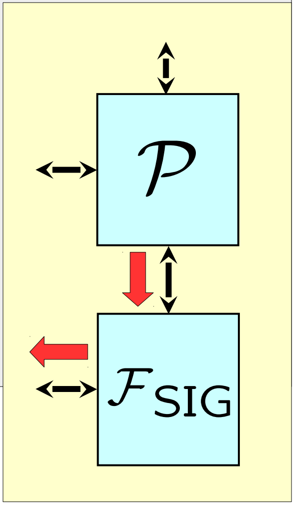
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

subroutine using urgent requests

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Problem propagates to higher level protocols:

Adversary is activated when calling a subroutine which models a local task.

The behavior of $\mathcal{P}$ in this case is undefined.

Examples from the literature

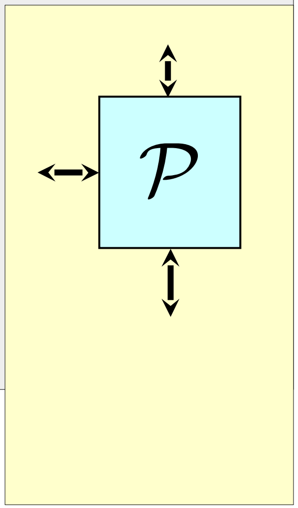
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$. **Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.

subroutine using urgent requests



Problem propagates to higher level protocols:

Adversary is activated when calling a subroutine which models a local task.

The behavior of $\mathcal{P}$ in this case is undefined.

Examples from the literature

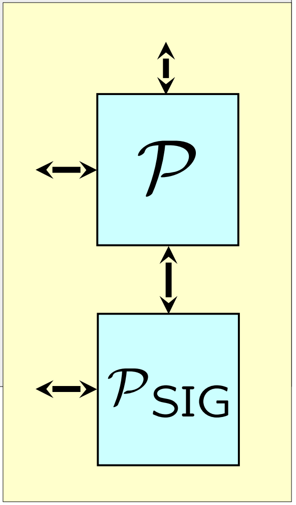
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

subroutine using urgent requests

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$. **Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Problem propagates to higher level protocols:

Adversary is activated when calling a subroutine which models a local task.

The behavior of $\mathcal{P}$ in this case is undefined.

Examples from the literature

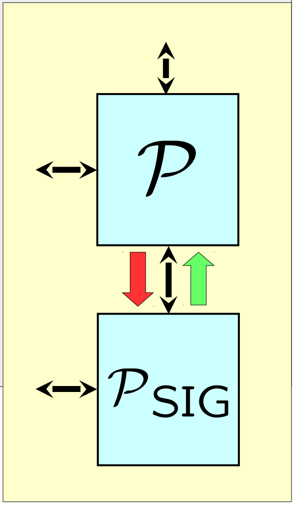
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

subroutine using urgent requests

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$. **Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



Problem propagates to higher level protocols:

Adversary is activated when calling a subroutine which models a local task.

The behavior of $\mathcal{P}$ in this case is undefined.

Examples from the literature

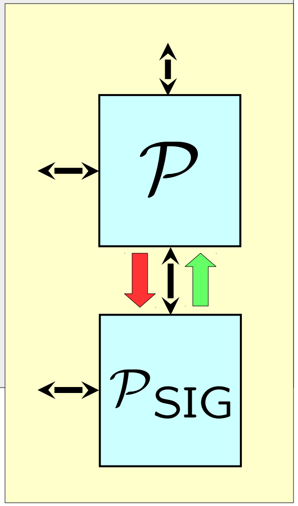
Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S sends $(\texttt{Sign}, (U, s), m)$ to $\mathcal{F}_{\text{SIG}}$. Upon receiving $(\texttt{Signature}, (U, s), m, \sigma)$ from $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.

subroutine using urgent requests



Idealization cannot express properties of realization:

Unlike $\mathcal{F}_{\text{SIG}}$, realization $\mathcal{P}_{\text{SIG}}$ is indeed local.

Problems from previous slides do not exist when using $\mathcal{P}_{\text{SIG}}$.

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests
- Code upload constructs

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests
- Code upload constructs
- Resort to a default

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests
- Code upload constructs
- Resort to a default

However:

- Not generally applicable

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests
- Code upload constructs
- Resort to a default

However:

- Not generally applicable
- Usually need tailor-made solutions

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests
- Code upload constructs
- Resort to a default

However:

- Not generally applicable
- Usually need tailor-made solutions
- Unnecessarily complicate
  specifications and proofs

Dealing with the Non-Responsiveness Problem

Workarounds for full specifications:

- Blocking requests while waiting for adversary
- Sending error messages while waiting for adversary
- Queuing new requests
- Code upload constructs
- Resort to a default

However:

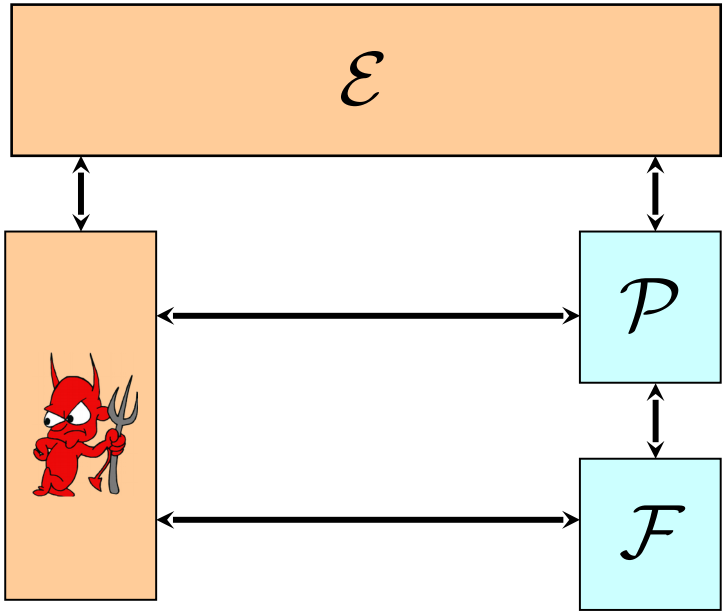
- Not generally applicable
- Usually need tailor-made solutions
- Unnecessarily complicate specifications and proofs

Also:

Does not address unintended state changes or limited expressivity

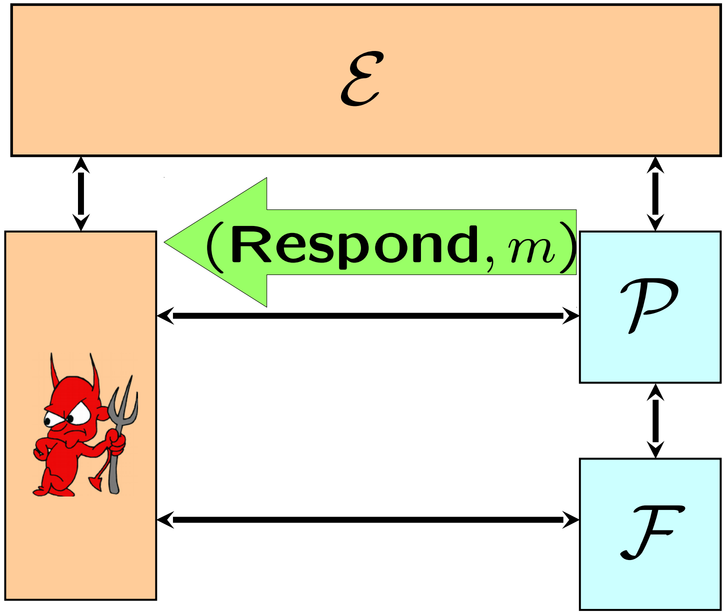
Our Solution

We introduce responsive environments and responsive adversaries



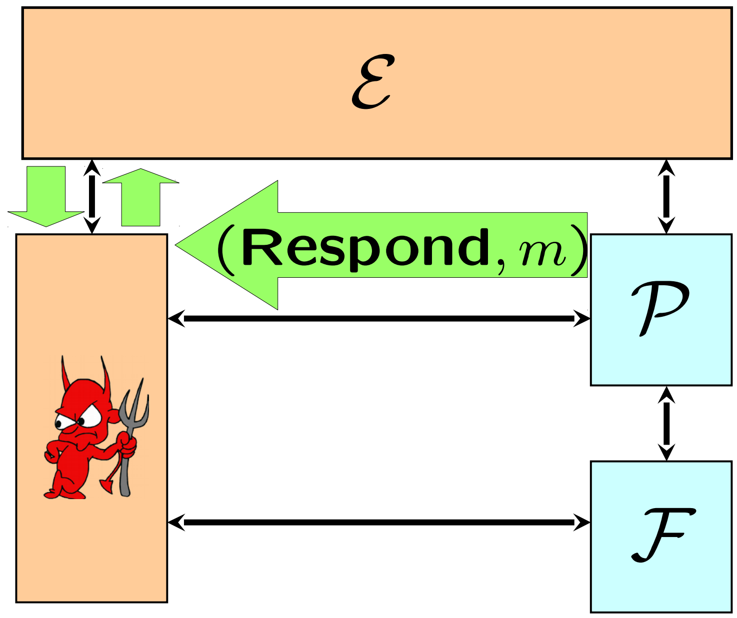
Our Solution

We introduce responsive environments and responsive adversaries



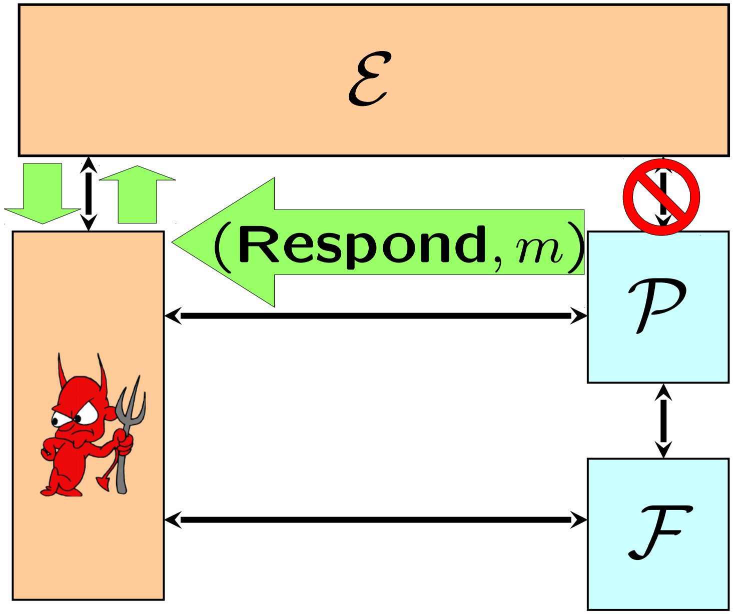
Our Solution

We introduce responsive environments and responsive adversaries



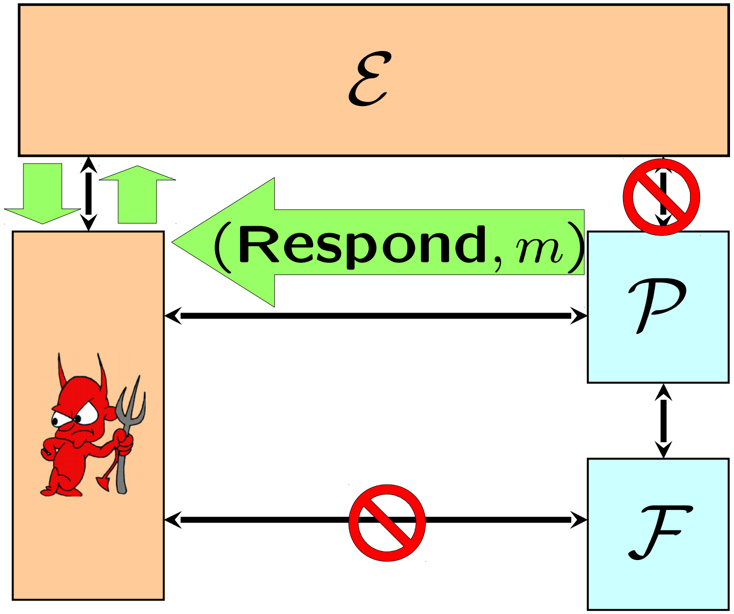
Our Solution

We introduce responsive environments and responsive adversaries



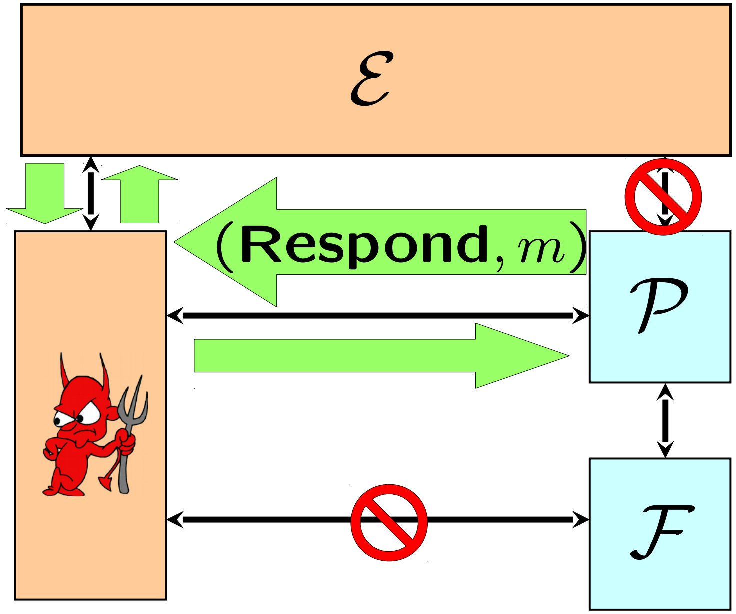
Our Solution

We introduce responsive environments and responsive adversaries



Our Solution

We introduce responsive environments and responsive adversaries



Non-Responsiveness Problem

Urgent requests do not model real network traffic

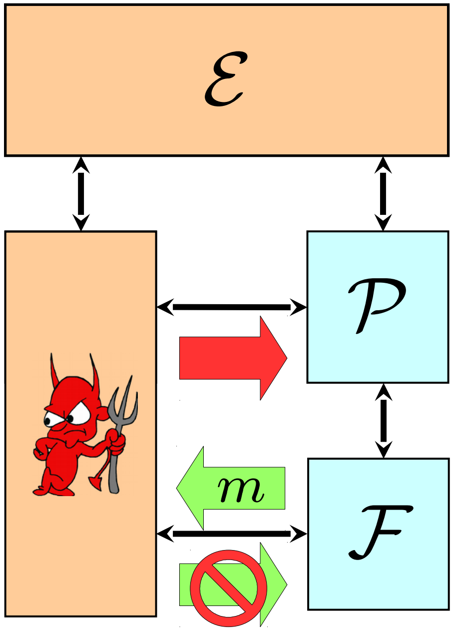
$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adversary can:

- Activate protocol in unexpected way
- Activate and change state of other parts of the protocol
- Block parts of the protocol



Non-Responsiveness Problem

Urgent requests do not model real network traffic

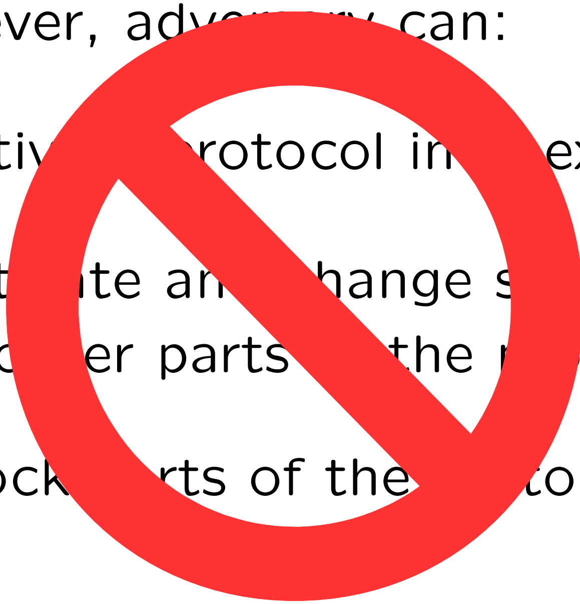
$\Rightarrow$ Real adversary cannot use them to mount attacks

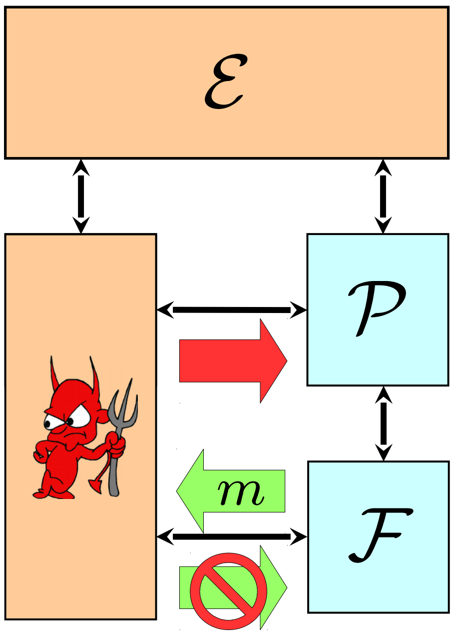
$\Rightarrow$ Natural to expect adversary in model to answer immediately

Non-Responsiveness Problem

However, adv… can:

- Activ… rotocol in… expected way
- Act… te an… hange s… te of o… er parts … the p… tocol
- Block … rts of the … tocol





Non-Responsiveness Problem

Urgent requests do not model real network traffic

$\Rightarrow$ Real adversary cannot use them to mount attacks

$\Rightarrow$ Natural to expect adversary in model to answer immediately

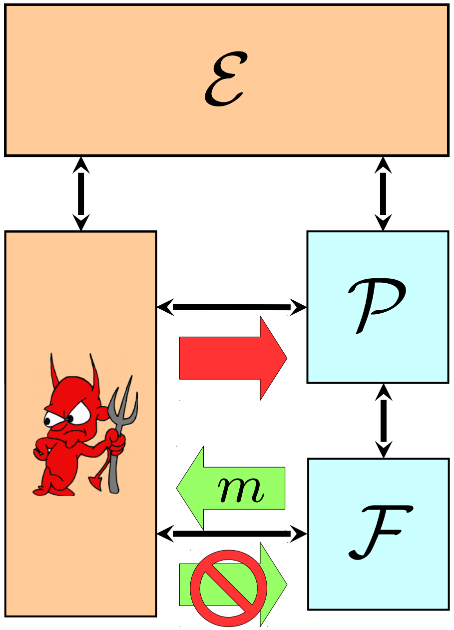
Non-Responsiveness Problem

However, ad ry can:

- Activ rotocol in expected way

- Act ate an hange s te
  of o er parts the tocol

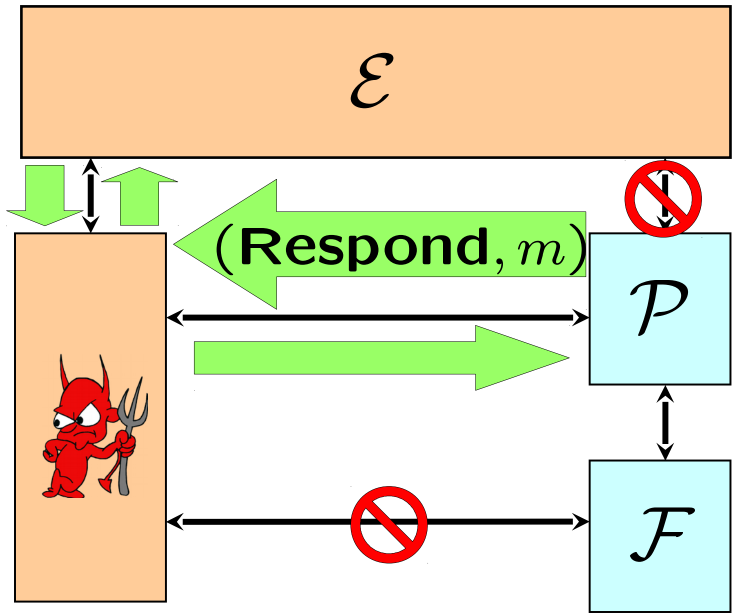
- Block rts of the tocol



Our Solution

We introduce responsive environments and responsive adversaries

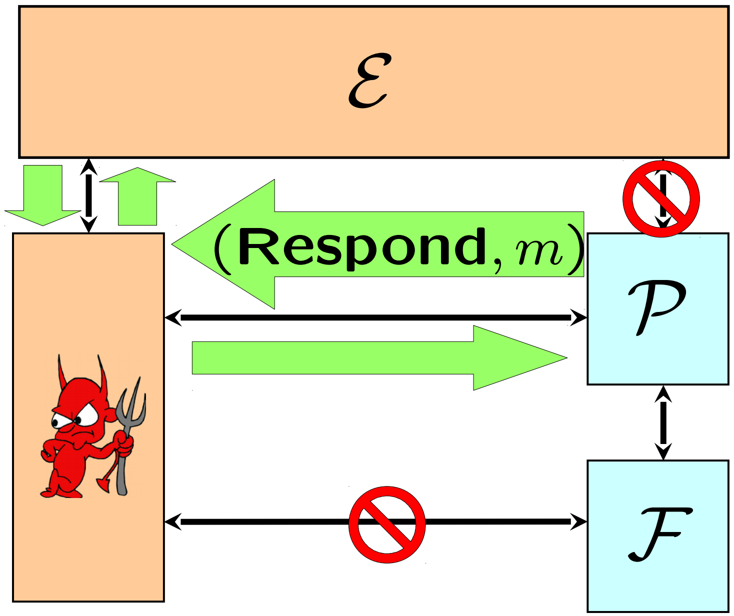
- Natural solution,
  solves the problem entirely



Our Solution

We introduce responsive environments and responsive adversaries

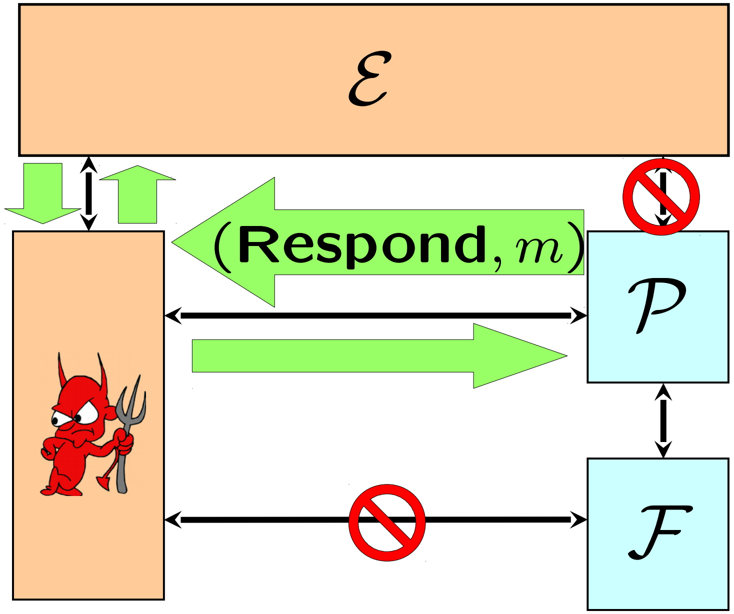
- Natural solution, solves the problem entirely
- Simple, elegant, easy to use



Our Solution

We introduce responsive environments and responsive adversaries

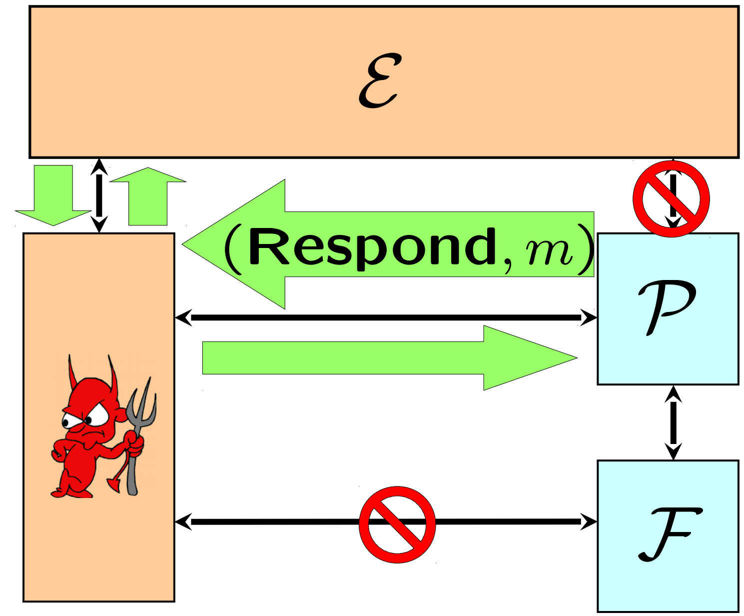
- Natural solution, solves the problem entirely
- Simple, elegant, easy to use
- Solves problems from the literature



Our Solution

We introduce responsive environments and responsive adversaries

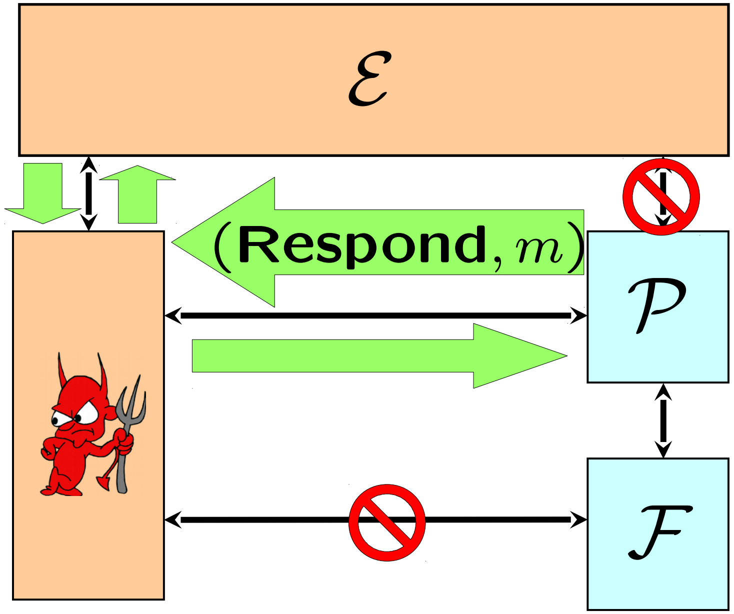
- Natural solution, solves the problem entirely
- Simple, elegant, easy to use
- Solves problems from the literature
- Applicable to all UC-style models (exemplified for UC, IITM, GNUC)



Our Solution

We introduce responsive environments and responsive adversaries

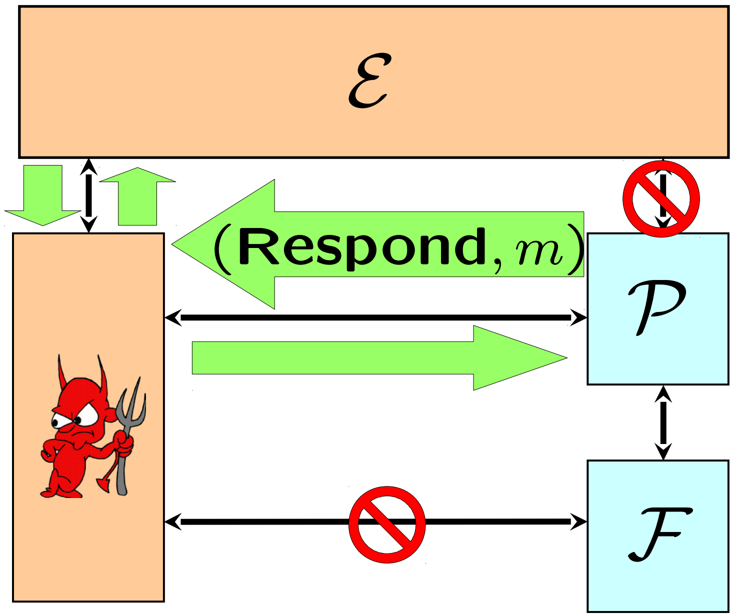
We provide detailed definitions and full proofs for the IITM model, including:



Our Solution

We introduce responsive environments and responsive adversaries

We provide detailed definitions and full proofs for the IITM model, including:



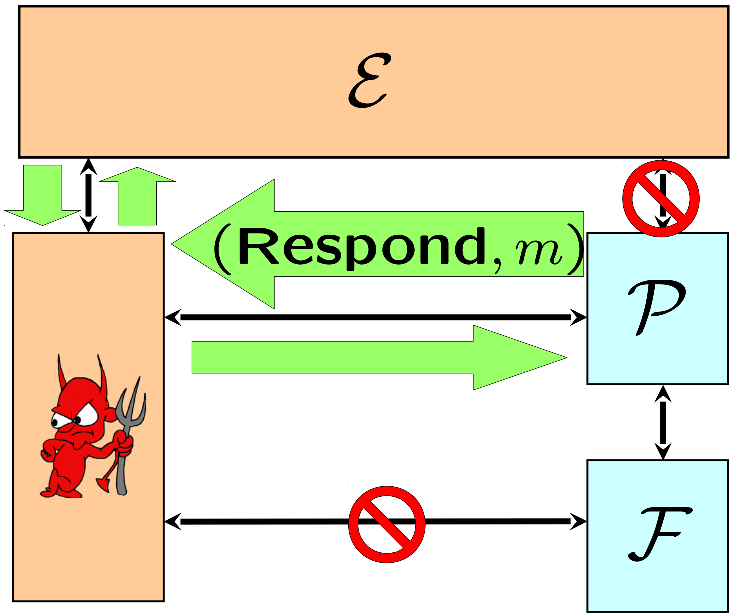
- Formal definitions of urgent requests, responsive environments, responsive adversaries

Our Solution

We introduce responsive environments and responsive adversaries

We provide detailed definitions and full proofs for the IITM model, including:

- Formal definitions of urgent requests, responsive environments, responsive adversaries
- Various security notions (dummy UC, strong simulatability, black-box simulatability, ...)

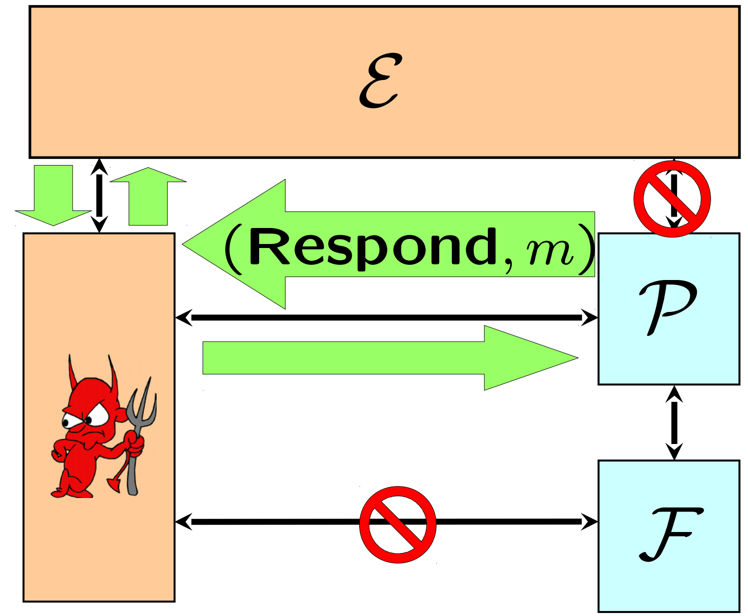


Our Solution

We introduce responsive environments and responsive adversaries

We provide detailed definitions and full proofs for the IITM model, including:

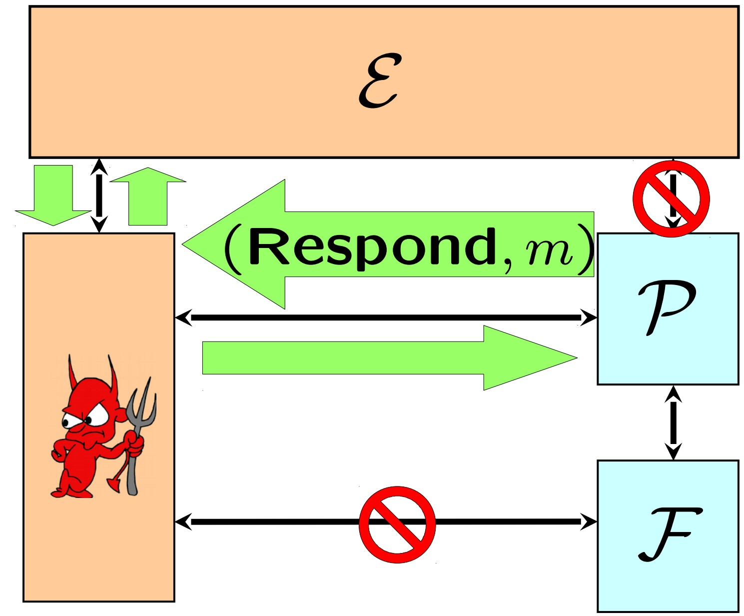
- Formal definitions of urgent requests, responsive environments, responsive adversaries
- Various security notions (dummy UC, strong simulatability, black-box simulatability, ...)
- Reflexivity and transitivity of security notions



Our Solution

We introduce responsive environments and responsive adversaries

We provide detailed definitions and full proofs for the IITM model, including:



- Formal definitions of urgent requests, responsive environments, responsive adversaries
- Various security notions (dummy UC, strong simulatability, black-box simulatability, ...)
- Reflexivity and transitivity of security notions
- Composition theorems

Solve problems from the literature

$\mathcal{F}_{\mathsf{NIKE}}$ from [Freire, Hesse, Hofheinz, 2014]

Upon input $(\texttt{init}, P_i, P_j)$ from P_i [...] consider two cases:

- Corrupted session mode: if there exists $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, **send** ($\texttt{init}, P_i, P_j)$ **to the adversary. After receiving** $(\{P_i, P_j\}, K_{i,j})$ **from the adversary**, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: [...]

Return (P_i, P_j, key) to P_i.

Solve problems from the literature

$\mathcal{F}_{\mathsf{NIKE}}$ from [Freire, Hesse, Hofheinz, 2014]

Upon input $(\mathtt{init}, P_i, P_j)$ from P_i [...] consider two cases:

- Corrupted session mode: if there exists $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, **send** $(\mathtt{Respond}, \mathtt{init}, P_i, P_j)$ **to the adversary. After receiving** $(\{P_i, P_j\}, K_{i,j})$ **from the adversary**, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: [...]

Return (P_i, P_j, key) to P_i.

Solve problems from the literature

$\mathcal{F}_{\mathsf{NIKE}}$ from [Freire, Hesse, Hofheinz, 2014]

Upon input $(\mathtt{init}, P_i, P_j)$ from P_i [...] consider two cases:

- Corrupted session mode: if there exists $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, **send** $(\mathtt{Respond}, \mathtt{init}, P_i, P_j)$ **to the adversary. After receiving** $(\{P_i, P_j\}, K_{i,j})$ **from the adversary**, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: [...]

Return (P_i, P_j, key) to P_i.

immediate response

Solve problems from the literature

$\mathcal{F}_{\text{NIKE}}$ from [Freire, Hesse, Hofheinz, 2014]

Upon input $(\texttt{init}, P_i, P_j)$ from P_i [...] consider two cases:

- Corrupted session mode: if there exists $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, **send** $(\texttt{Respond}, \texttt{init}, P_i, P_j)$ **to the adversary. After receiving** $(\{P_i, P_j\}, K_{i,j})$ **from the adversary**, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: [...]

Return (P_i, P_j, key) to P_i.

immediate response

Solve problems from the literature

$\mathcal{F}_{\mathsf{sok}}$ from [Chase, Lysyanskaya, 2006]

Upon receiving a value $(\mathtt{Setup}, sid)$ from any party P, verify that $sid = (M_L, sid')$ for some sid'. If not, then ignore the request. Else, if this is the first time that $(\mathtt{Setup}, sid)$ was received, **hand** $(\mathtt{Respond}, \mathtt{Setup}, sid)$ **to the adversary; upon receiving** $(\mathtt{Algorithms}, sid, \text{Verify}, \text{Sign}, \text{Simsign}, \text{Extract})$ **from the adversary**, store these algorithms. Output the stored $(\text{Algorithms}, sid, \text{Sign}, \text{Verify})$ to P.

Solve problems from the literature

$\mathcal{F}_{\text{sok}}$ from [Chase, Lysyanskaya, 2006]

Upon receiving a value $(\texttt{Setup}, sid)$ from any party P, verify that $sid = (M_L, sid')$ for some sid'. If not, then ignore the request. Else, if this is the first time that $(\texttt{Setup}, sid)$ was received, **hand** $(\texttt{Respond}, \texttt{Setup}, sid)$ **to the adversary; upon receiving** $(\texttt{Algorithms}, sid, \text{Verify}, \text{Sign}, \text{Simsign}, \text{Extract})$ **from the adversary**, store these algorithms. Output the stored $(\text{Algorithms}, sid, \text{Sign}, \text{Verify})$ to P.

immediate response

Solve problems from the literature

$\mathcal{F}_{\mathsf{sok}}$ from [Chase, Lysyanskaya, 2006]

Upon receiving a value $(\mathtt{Setup}, sid)$ from any party P, verify that $sid = (M_L, sid')$ for some sid'. If not, then ignore the request. Else, if this is the first time that $(\mathtt{Setup}, sid)$ was received, **hand** $(\mathtt{Respond}, \mathtt{Setup}, sid)$ **to the adversary; upon receiving** $(\mathtt{Algorithms}, sid, \text{Verify}, \text{Sign}, \text{Simsign}, \text{Extract})$ **from the adversary**, store these algorithms. Output the stored $(\text{Algorithms}, sid, \text{Sign}, \text{Verify})$ to P.

immediate response

Solve problems from the literature

$\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Upon receiving a value $(\mathtt{Verify}, sid, m, \sigma)$ from some party S', **hand** $(\mathtt{Respond}, \mathtt{Verify}, sid, m, \sigma)$ **to the adversary. Upon receiving** $(\mathtt{Verified}, sid, m, \phi)$ **from the adversary**, do:

[...]

Output $(\mathtt{Verified}, sid, m, f)$ to S'.

Solve problems from the literature

$\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Upon receiving a value $(\texttt{Verify}, sid, m, \sigma)$ from some party S', **hand** $(\texttt{Respond}, \texttt{Verify}, sid, m, \sigma)$ **to the adversary. Upon receiving** $(\texttt{Verified}, sid, m, \phi)$ **from the adversary**, do:

[…]

Output $(\texttt{Verified}, sid, m, f)$ to S'.

immediate response

Solve problems from the literature

$\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Upon receiving a value $(\texttt{Verify}, sid, m, \sigma)$ from some party S', **hand** $(\texttt{Respond}, \texttt{Verify}, sid, m, \sigma)$ **to the adversary. Upon receiving** $(\texttt{Verified}, sid, m, \phi)$ **from the adversary**, do:
[...]
Output $(\texttt{Verified}, sid, m, f)$ to S'.

immediate response

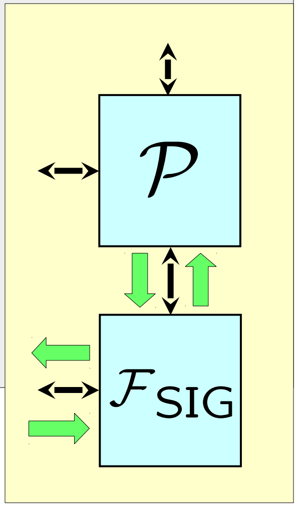
Solve problems from the literature

Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



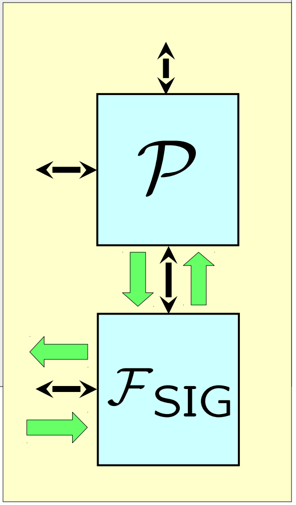
Solve problems from the literature

Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



signature returned immediately

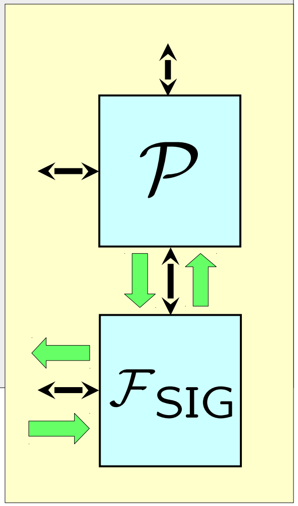
Solve problems from the literature

Realization of $\mathcal{F}_{\text{D-Cert}}$ from [Zhao, Zhang, Qin, Feng, 2014]

Signature Protocol: When activated with input $(\texttt{Sign}, sid, m)$, Party S does:

[...]

S **sends** $(\texttt{Sign}, (U, s), m)$ **to** $\mathcal{F}_{\text{SIG}}$**. Upon receiving** $(\texttt{Signature}, (U, s), m, \sigma)$ **from** $\mathcal{F}_{\text{SIG}}$, S outputs $(\texttt{Signature}, sid, m, \sigma)$.



signature returned immediately

Conclusion

- Protocols often exchange meta information by what we call urgent requests

Conclusion

- Protocols often exchange meta information by what we call urgent requests

- Non-Responsiveness Problem:
  Adversary might not answer immediately

Conclusion

- Protocols often exchange meta information by what we call urgent requests

- Non-Responsiveness Problem:
  Adversary might not answer immediately
  * Complicates protocol specifications and security proofs

Conclusion

- Protocols often exchange meta information by what we call urgent requests

- Non-Responsiveness Problem:
  Adversary might not answer immediately
  * Complicates protocol specifications and security proofs
  * No simple, general solution for adjusting protocols

Conclusion

- Protocols often exchange meta information by what we call urgent requests

- Non-Responsiveness Problem:
  Adversary might not answer immediately
  * Complicates protocol specifications and security proofs
  * No simple, general solution for adjusting protocols
  * Limited expressiveness

Conclusion

- Protocols often exchange meta information by what we call urgent requests

- Non-Responsiveness Problem:
  Adversary might not answer immediately
  * Complicates protocol specifications and security proofs
  * No simple, general solution for adjusting protocols
  * Limited expressiveness
  * Often ignored in the literature

Conclusion

- Protocols often exchange meta information by what we call urgent requests

- Non-Responsiveness Problem:
  Adversary might not answer immediately
  * Complicates protocol specifications and security proofs
  * No simple, general solution for adjusting protocols
  * Limited expressiveness
  * Often ignored in the literature
    - Underspecified protocols
    - Flawed proofs
    - Hard to reuse functionalities

Conclusion

- Our solution: Responsive environments/adversaries
  Easy to use, gets rid of the problem entirely, fixes literature

Conclusion

- Our solution: Responsive environments/adversaries
  Easy to use, gets rid of the problem entirely, fixes literature

Use our framework!

It makes your life much easier!

Conclusion

- Our solution: Responsive environments/adversaries
  Easy to use, gets rid of the problem entirely, fixes literature

Use our framework!

It makes your life much easier!

Thanks for your attention!