MoSS:

Modular Security Specifications Framework

Amir Herzberg[1], Hemi Leibowitz[2], Ewa Syta[3], **Sara Wrótniak**[1]

[1]University of Connecticut [2]Bar-Ilan University, Israel [3]Trinity College

Crypto 2021

Provable Security for Applied Protocols

- Proofs are critical to ensure security
- Many applied protocols are not proven secure

Provable Security for Applied Protocols

- Proofs are critical to ensure security
- Many applied protocols are not proven secure
- Admittedly, proving security of applied protocols is challenging
  - Easy to make subtle mistakes
  - Applied protocols and specifications can be complex
  - Reflecting real-world environment assumptions is non-trivial

Approaches to Protocol Specifications

Approach	Models (assumptions)	Requirements (goals)	Provable security	Provably secure composition
Informal	List of descriptions	List of descriptions	No	No

Approaches to Protocol Specifications

Approach	Models (assumptions)	Requirements (goals)	Provable security	Provably secure composition
Informal	List of descriptions	List of descriptions	No	No

- Protocol specifications include assumptions and goals

Approaches to Protocol Specifications

Approach	Models (assumptions)	Requirements (goals)	Provable security	Provably secure composition
Informal	List of descriptions	List of descriptions	No	No

- Protocol specifications include assumptions and goals
- *Models* are assumptions, including adversary (e.g., MitM), communication (e.g., reliable), synchronization (e.g., bounded-drift clocks), initialization (e.g, shared keys)
- *Requirements* are goals, including generic (e.g, indistinguishability) and specific (e.g., authenticated broadcast)

Approaches to Protocol Specifications

Approach	Models (assumptions)	Requirements (goals)	Provable security	Provably secure composition
Informal	List of descriptions	List of descriptions	No	No
Game-based	Game per goal; models are part of game		Yes	No

- *Models* are assumptions, including adversary (e.g., MitM), communication (e.g., reliable), synchronization (e.g., bounded-drift clocks), initialization (e.g, shared keys)
- *Requirements* are goals, including generic (e.g, indistinguishability) and specific (e.g., authenticated broadcast)

Approaches to Protocol Specifications

Approach	Models (assumptions)	Requirements (goals)	Provable security	Provably secure composition
Informal	List of descriptions	List of descriptions	No	No
Game-based	Game per goal; models are part of game		Yes	No
Simulation-based	Indistinguishable from Ideal Functionality		Yes	Yes

- *Models* are assumptions, including adversary (e.g., MitM), communication (e.g., reliable), synchronization (e.g., bounded-drift clocks), initialization (e.g, shared keys)
- *Requirements* are goals, including generic (e.g, indistinguishability) and specific (e.g., authenticated broadcast)

Approaches to Protocol Specifications

Approach	Models (assumptions)	Requirements (goals)	Provable security	Provably secure composition
Informal	List of descriptions	List of descriptions	No	No
Game-based	Game per goal; models are part of game		Yes	No
Simulation-based	Indistinguishable from Ideal Functionality		Yes	Yes
MoSS	List of predicates	List of predicates	Yes	No

- **MoSS** *models* are **well-defined** assumptions, including adversary (e.g., MitM), communication (e.g., reliable), synchronization (e.g., bounded-drift clocks), initialization (e.g, shared keys)
- **MoSS** *requirements* are **well-defined** goals, including generic (e.g, indistinguishability) and specific (e.g., authenticated broadcast)

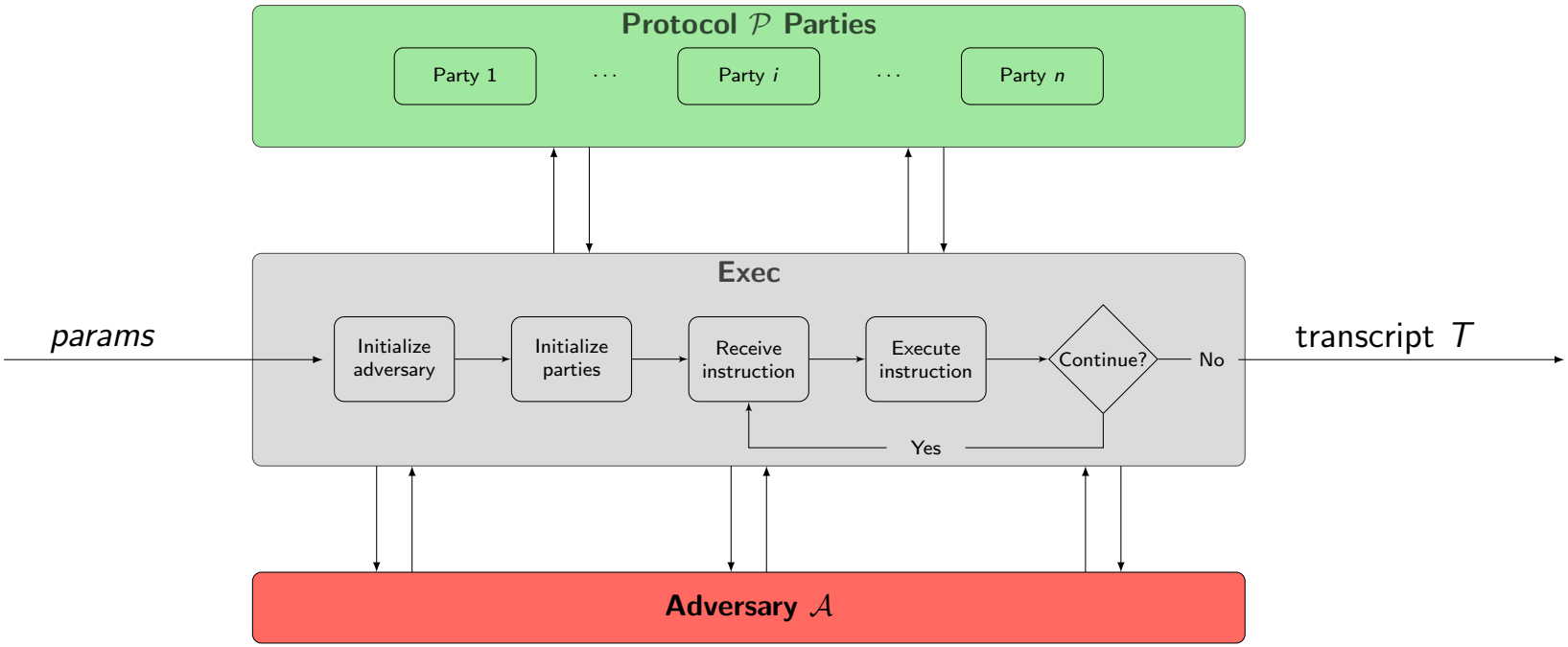
What is MoSS?

- Modular Security Specifications Framework
  - Game-based, with well-defined specifications
- MoSS **separates** the execution process from the *model* under which a protocol is analyzed and the *requirements* that define the protocol's goals.

What is MoSS?

- Modular Security Specifications Framework
  - Game-based, with well-defined specifications
- MoSS **separates** the execution process from the *model* under which a protocol is analyzed and the *requirements* that define the protocol's goals.
- This separation allows:
  - Gradual development of specifications
  - Reusability of definitions and results
  - Simplified and concise analysis

MoSS's Adversary-Driven Execution Process



- Adversary $\mathcal{A}$ and protocol $\mathcal{P}$ are represented as stateless functions
- Execution returns execution transcript T

MoSS's Adversary-Driven Execution Process

1: $(s_{\mathcal{A}}, \mathsf{N}, params.\mathcal{P}[\cdot]) \leftarrow \mathcal{A}[\text{'Init'}](params)$

2: $\forall i \in \mathsf{N}: \; s_i \leftarrow \mathcal{P}[\text{'Init'}](\bot, params.\mathcal{P}[i], \bot)$

3: $e \leftarrow 0$

4: **repeat**

5: $\quad e \leftarrow e + 1$

6: $\quad (ent[e], opr[e], inp[e], clk[e], \tau[e]) \leftarrow \mathcal{A}(s_{\mathcal{A}})$

7: $\quad s^{In}[e] \leftarrow s_{ent[e]}$

8: $\quad (s_{ent[e]}, out[e]) \leftarrow \mathcal{P}[opr[e]](s_{ent[e]}, inp[e], clk[e])$

9: $\quad s^{Out}[e] \leftarrow s_{ent[e]}$

10: $\quad (s_{\mathcal{A}}, out_{\mathcal{A}}, \mathsf{F}) \leftarrow \mathcal{A}(s_{\mathcal{A}}, out[e])$

11: **until** $out_{\mathcal{A}} \neq \bot$

12: $T \leftarrow (out_{\mathcal{A}}, e, \mathsf{N}, \mathsf{F}, ent[\cdot], opr[\cdot], inp[\cdot], clk[\cdot], \tau[\cdot], out[\cdot], params.\mathcal{P}[\cdot], s^{In}[\cdot], s^{Out}[\cdot])$

13: Return T

MoSS's Adversary-Driven Execution Process

1: $(s_{\mathcal{A}}, \mathsf{N}, params.\mathcal{P}[\cdot]) \leftarrow \mathcal{A}[\text{'Init'}](params)$

2: $\forall i \in \mathsf{N}: \; s_i \leftarrow \mathcal{P}[\text{'Init'}](\bot, params.\mathcal{P}[i], \bot)$

3: $e \leftarrow 0$

4: **repeat**

5: $\quad e \leftarrow e + 1$

6: $\quad (ent[e], opr[e], inp[e], clk[e], \tau[e]) \leftarrow \mathcal{A}(s_{\mathcal{A}})$

7: $\quad s^{In}[e] \leftarrow s_{ent[e]}$

8: $\quad (s_{ent[e]}, out[e]) \leftarrow \mathcal{P}[opr[e]](s_{ent[e]}, inp[e], clk[e])$

9: $\quad s^{Out}[e] \leftarrow s_{ent[e]}$

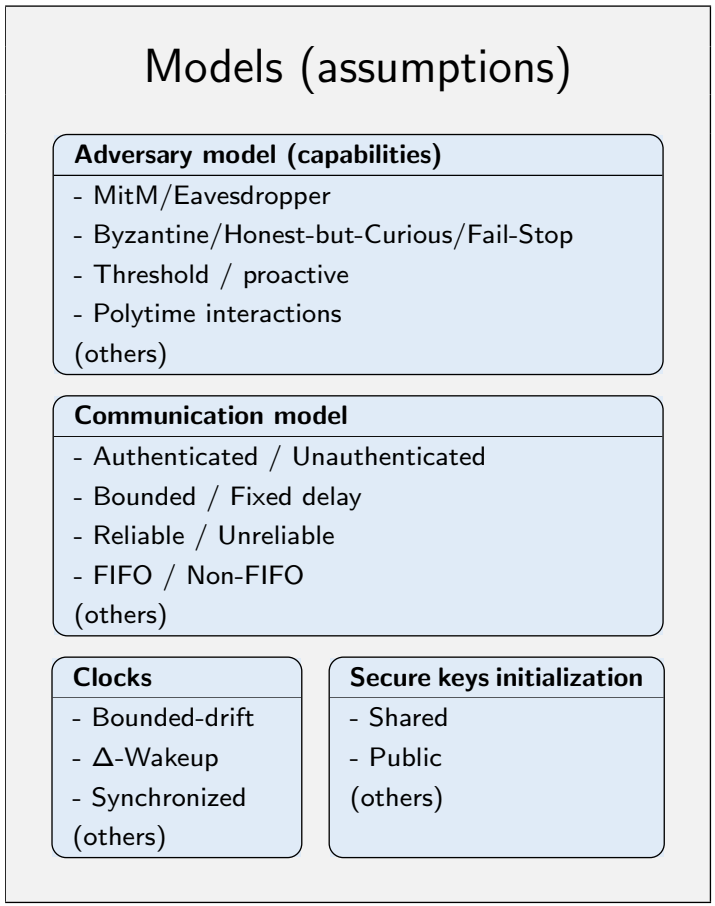
10: $\quad (s_{\mathcal{A}}, out_{\mathcal{A}}, \mathsf{F}) \leftarrow \mathcal{A}(s_{\mathcal{A}}, out[e])$

11: **until** $out_{\mathcal{A}} \neq \bot$

12: $T \leftarrow (out_{\mathcal{A}}, e, \mathsf{N}, \mathsf{F}, ent[\cdot], opr[\cdot], inp[\cdot], clk[\cdot], \tau[\cdot], out[\cdot], params.\mathcal{P}[\cdot], s^{In}[\cdot], s^{Out}[\cdot])$

13: Return T

Models



- Check if an adversary **satisfies** a model: $\mathcal{A} \models_{\text{poly}} \mathcal{M}$
- Models are independent
- Can be reused across systems

Models

- A model is a set of pairs (π, β) which we call *specifications*, where π is a predicate and β is the *base function*, which specifies the probability which is allowed for the adversary to win against π

Models

- A model is a set of pairs (π, β) which we call *specifications*, where π is a predicate and β is the *base function*, which specifies the probability which is allowed for the adversary to win against π

- The base function may be, e.g., $\frac{1}{2}$ for confidentiality properties, 2^{-l} for authentication with l-bit tags, 0 for others

Models

- A model is a set of pairs (π, β) which we call *specifications*, where π is a predicate and β is the *base function*, which specifies the probability which is allowed for the adversary to win against π

- The base function may be, e.g., $\frac{1}{2}$ for confidentiality properties, 2^{-l} for authentication with l-bit tags, 0 for others

Definition ($\mathcal{A}$ satisfies $\mathcal{M} = \{(\pi, \beta)\}$)

Adversary $\mathcal{A}$ satisfies model $\mathcal{M} = \{(\pi, \beta)\}$, denoted $\mathcal{A} \models_{\text{poly}} \mathcal{M}$, if for every protocol $\mathcal{P}$, the probability that $\pi(T, \mathit{params}) = \perp$ is at most negligibly greater than $\beta(\mathit{params})$, where $T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(\mathit{params})$.

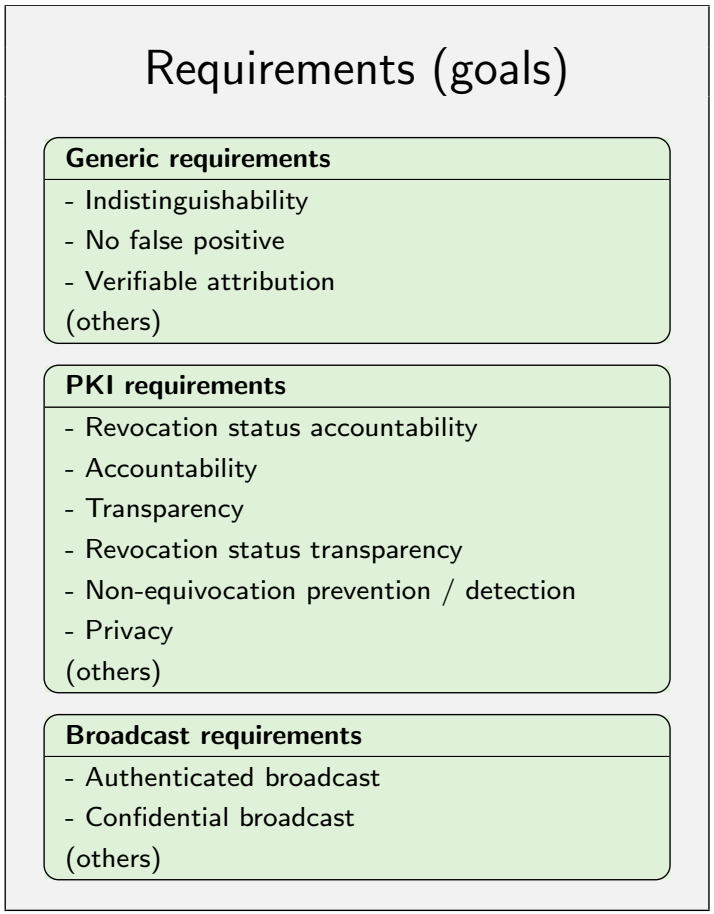
Example Model

- Bounded clock drift model $\mathcal{M}^{\text{Drift}}_{\Delta_{clk}} = \{(\pi^{\text{Drift}}_{\Delta_{clk}}, 0)\}$

Algorithm 1 Bounded clock drift model predicate $\pi^{\text{Drift}}_{\Delta_{clk}}(T, params)$

1: **return** (
2: $\quad \forall \hat{e} \in \{1, \ldots, T.e\}$: ▷ *For each event*
3: $\quad\quad |T.clk[\hat{e}] - T.\tau[\hat{e}]| \leq \Delta_{clk}$ ▷ *Local clock is within Δ_{clk} drift from real time*
4: $\quad\quad$ **and if** $\hat{e} \geq 2$ **then** $T.\tau[\hat{e}] \geq T.\tau[\hat{e}-1]$ ▷ *In each consecutive event, the real time difference is monotonically increasing*
)

Requirements



- Check if a protocol satisfies a **requirement** under a **model**: $\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R}$
- Requirements are well-defined using predicates
- Comparable

Requirements

- A requirement is a set of pairs (π, β) which we call *specifications*, where π is a predicate and β is the *base function*, which specifies the probability which is allowed for the adversary to win against π

Requirements

- A requirement is a set of pairs (π, β) which we call *specifications*, where π is a predicate and β is the *base function*, which specifies the probability which is allowed for the adversary to win against π

Definition ($\mathcal{P}$ satisfies $\mathcal{R} = \{(\pi, \beta)\}$ under $\mathcal{M}$)

Protocol $\mathcal{P}$ satisfies requirement $\mathcal{R} = \{(\pi, \beta)\}$ under model $\mathcal{M}$, denoted $\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R}$, if for every adversary $\mathcal{A}$ which satisfies $\mathcal{M}$, the probability that $\pi(T, \mathit{params}) = \perp$ is at most negligibly greater than $\beta(\mathit{params})$, where $T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(\mathit{params})$.

Example Requirement

- Δ-transparency requirement $\mathcal{R}_{\Delta\text{TRA}} = \{(\pi_{\Delta\text{TRA}}, 0)\}$

Intuitive definition (Δ-transparecy)

A certificate attested as Δ-transparent must be available to all 'interested' parties within Δ time of its transparency attestation being issued by a proper authority.

$$\pi_{\Delta\text{TRA}}(T, params) \equiv \left\{ \begin{array}{l} (\psi, \rho, pk, \iota, \iota_M) \leftarrow T.out_{\mathcal{A}}; \\ \textbf{return } \neg \left[\begin{array}{l} \textsc{HonestEntity}(T, params, \iota) \wedge \\ \textsc{CorrectPublicKey}(T, params, \iota, pk, \rho.\iota) \wedge \\ \textsc{ValidCertificateAttestation}(T, params, \{\Delta\text{TRA}\}, \psi, pk, \rho) \wedge \\ \textsc{HonestEntity}(T, params, \iota_M) \wedge \\ \textsc{IsMonitor}(T, params, \iota_M, \rho.\iota) \wedge \\ \textsc{HonestMonitorUnawareOfCertificate}(T, params) \wedge \\ \textsc{WasNotAccused}(T, params) \end{array} \right]; \end{array} \right\}$$

Modularity Lemmas

- Formalize intuitive modularity properties of MoSS models and requirements

Modularity Lemmas

- Formalize intuitive modularity properties of MoSS models and requirements

Lemma (Requirement-model monotonicity lemma)

For any models $\mathcal{M}, \widehat{\mathcal{M}}$ where $\mathcal{M} \subseteq \widehat{\mathcal{M}}$ and any requirement $\mathcal{R}$:

$$\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R} \Rightarrow \mathcal{P} \models^{\widehat{\mathcal{M}}}_{\text{poly}} \mathcal{R}$$

Modularity Lemmas

- Formalize intuitive modularity properties of MoSS models and requirements

Lemma (Requirement-model monotonicity lemma)

For any models $\mathcal{M}, \widehat{\mathcal{M}}$ where $\mathcal{M} \subseteq \widehat{\mathcal{M}}$ and any requirement $\mathcal{R}$:

$$\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R} \Rightarrow \mathcal{P} \models^{\widehat{\mathcal{M}}}_{\text{poly}} \mathcal{R}$$

Lemma (Requirements union lemma)

For any models $\mathcal{M}, \mathcal{M}'$ and any requirements $\mathcal{R}, \mathcal{R}'$, let $\widehat{\mathcal{M}} \equiv \mathcal{M} \cup \mathcal{M}'$ and $\widehat{\mathcal{R}} \equiv \mathcal{R} \cup \mathcal{R}'$. Then:

$$\left(\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R} \wedge \mathcal{P} \models^{\mathcal{M}'}_{\text{poly}} \mathcal{R}'\right) \Rightarrow \mathcal{P} \models^{\widehat{\mathcal{M}}}_{\text{poly}} \widehat{\mathcal{R}}$$

Modularity Lemmas

- Formalize intuitive modularity properties of MoSS models and requirements

Lemma (Requirement-model monotonicity lemma)

For any models $\mathcal{M}, \widehat{\mathcal{M}}$ where $\mathcal{M} \subseteq \widehat{\mathcal{M}}$ and any requirement $\mathcal{R}$:

$$\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R} \Rightarrow \mathcal{P} \models^{\widehat{\mathcal{M}}}_{\text{poly}} \mathcal{R}$$

Lemma (Requirements union lemma)

For any models $\mathcal{M}, \mathcal{M}'$ and any requirements $\mathcal{R}, \mathcal{R}'$, let $\widehat{\mathcal{M}} \equiv \mathcal{M} \cup \mathcal{M}'$ and $\widehat{\mathcal{R}} \equiv \mathcal{R} \cup \mathcal{R}'$. Then:

$$\left(\mathcal{P} \models^{\mathcal{M}}_{\text{poly}} \mathcal{R} \wedge \mathcal{P} \models^{\mathcal{M}'}_{\text{poly}} \mathcal{R}'\right) \Rightarrow \mathcal{P} \models^{\widehat{\mathcal{M}}}_{\text{poly}} \widehat{\mathcal{R}}$$

- See paper for more lemmas

Modularity and Gradual Analysis

- The use of separate, focused models and requirements also allows a *gradual protocol development and analysis*.

Modularity and Gradual Analysis

- The use of separate, focused models and requirements also allows a *gradual protocol development and analysis*.
- Let $\mathcal{P}$ be an authenticated-broadcast protocol.

Modularity and Gradual Analysis

- The use of separate, focused models and requirements also allows a *gradual protocol development and analysis*.
- Let $\mathcal{P}$ be an authenticated-broadcast protocol.
  1. Prove that $\mathcal{P}$ ensures authenticity assuming a secure shared-key initialization model.

Modularity and Gradual Analysis

- The use of separate, focused models and requirements also allows a *gradual protocol development and analysis*.
- Let $\mathcal{P}$ be an authenticated-broadcast protocol.
  1. Prove that $\mathcal{P}$ ensures authenticity assuming a secure shared-key initialization model.
  2. Then, show that $\mathcal{P}$ also achieves freshness when we also assume bounded clock drift.

Modularity and Gradual Analysis

- The use of separate, focused models and requirements also allows a *gradual protocol development and analysis*.

- Let $\mathcal{P}$ be an authenticated-broadcast protocol.

  1. Prove that $\mathcal{P}$ ensures authenticity assuming a secure shared-key initialization model.
  2. Then, show that $\mathcal{P}$ also achieves freshness when we also assume bounded clock drift.
  3. Lastly, show that $\mathcal{P}$ ensures a bounded delay by additionally assuming bounded-delay communication.

MoSS Can Do Much More

- Extendable execution process
  - Entity corruptions, confidentiality, shared keys, etc.
- Built-in support for concrete security
  - Important to use concrete parameter values in practice
- Allows to ensure polynomial-time execution
  - Not trivial due to interaction, unlimited invocations, state

Future Work

- Modular specifications and compositions
  - Both modular specifications and modular protocol design, i.e., provably-secure compositions
- Modular simulation-based specifications
  - Modular specifications in simulation-based framework(s) and/or support for simulation-based specifications in MoSS
- Computer-aided analysis
  - Translating MoSS specifications to other forms and/or developing tools for MoSS directly

Conclusions

- MoSS facilitates modular security specifications of cryptographic and other protocols
- Allows provable security for practical crypto/security protocols, with complex models (delays, faults, etc.)
- Allows reuse of models, requirements
- Paper contains details, proofs and more, e.g., concrete-security