

Robust Logical Foundations for Mechanizing Post-Quantum Cryptography in Squirrel

FCS 2026

Adrien Koutsos Inria Paris

Joint work with:

David Baelde Univ Rennes, IRISA, CNRS

Antoine Dallon AMIAD

Stéphanie Delaune Univ Rennes, IRISA, CNRS

Charlie Jacomme Inria Nancy

July 25 2026, Lisbon, Portugal



Context

Cryptographic protocols are critical.



Provable Security

Cryptographic protocols are **critical**.



Computer-Aided Cryptography (CAC)

- **Provable security**: mathematical proofs of security.
- **Mechanization**: high level of confidence.

$$\forall \text{cat} \in \mathcal{C}. \Pr(\text{cat} \text{ breaks } \mathcal{P}) \leq \epsilon$$

Provable Security

Cryptographic protocols are **critical**.



Computer-Aided Cryptography (CAC)

- **Provable security**: mathematical proofs of security.
- **Mechanization**: high level of confidence.

$$\forall \text{cat} \in \mathcal{C}. \Pr(\text{cat} \text{ breaks } \mathcal{P}) \leq \epsilon$$

Attacker model: $\mathcal{C} = \text{PPTM}$

Polynomial-time Probabilistic Turing Machine

CAC frameworks: CryptoVerif, **Squirrel**, EasyCrypt, SSProve

Attacker model: capture realistic attack scenario

Attacker model: capture realistic attack scenario

Quantum Computers

- Quantum computers (QC) on their way (?)
- QC breaks many existing crypto systems
Discrete logarithm, Diffie-Hellman, ...

Attacker model: capture realistic attack scenario

Quantum Computers

- Quantum computers (QC) on their way (?)
- QC breaks many existing crypto systems
Discrete logarithm, Diffie-Hellman, ...

Post-Quantum Cryptography (PQC)

Secure cryptography against quantum adversaries:
quantum adversary, **classical** protocol.

PQC effort: ML-KEM, ML-DSA, Signal (PQXDH, SPQR), iMessage (PQ3)

CAC for PQC (*work-in-progress*)

Mechanized cryptographic proofs of PQ security.

$$\forall \mathfrak{c} \in \mathcal{C}. \Pr(\mathfrak{c} \text{ breaks } \mathcal{P}) \leq \epsilon \quad (\mathcal{C} = \text{PQTM})$$

PQC Frameworks: [CryptoVerif](#), **PQ-Squirrel**, [EasyPQC](#), [qrhl-tool](#)

Limitations of PQ-Squirrel [S&P'22]

■ Expressivity:

Capture PQTMs using **black-box interactive machines**

⇒ quantum values not represented (e.g. no QROM)

■ Maintainability:

- Unusual **stateful** semantics
- Distance with implem.
- Lacks latest improvements (theory, implem.), e.g. **crypto**, **smt**

Limitations of PQ-Squirrel [S&P'22]

■ Expressivity:

Capture PQTMs using **black-box interactive machines**

⇒ quantum values not represented (e.g. no QROM)

■ Maintainability:

- Unusual **stateful** semantics
- Distance with implem.
- Lacks latest improvements (theory, implem.), e.g. **crypto**, **smt**

Goal

Design a **robust** PQ extension of Squirrel

Cryptographic Reductions

$$\forall c \in \mathcal{C}. \Pr(c \text{ breaks } \mathcal{P}) \leq \epsilon$$

Cryptographic Reductions

$$\forall \text{cat} \in \mathcal{C}. \Pr(\text{cat} \text{ breaks } \mathcal{P}) \leq \epsilon$$

Cryptographic reductions: proof technique for provable security

Similar to theoretical complexity reductions (e.g. **Hamilton** $\Rightarrow$ **SAT**)

$$(\exists \text{cat} \in \mathcal{C} \text{ breaking } \mathcal{P}) \Rightarrow (\exists \text{cat} \in \mathcal{C} \text{ breaking } \mathcal{H})$$

Hardness assumption $\mathcal{H}$ cannot be broken by any $\text{cat} \in \mathcal{C}$

Examples: Discrete-Log, IND-CPA, ...

Cryptographic Reductions

$$\forall \text{cat} \in \mathcal{C}. \Pr(\text{cat} \text{ breaks } \mathcal{P}) \leq \epsilon$$

Cryptographic reductions: proof technique for provable security

Similar to theoretical complexity reductions (e.g. **Hamilton** $\Rightarrow$ **SAT**)

$$(\exists \text{cat} \in \mathcal{C} \text{ breaking } \mathcal{P}) \Rightarrow (\exists \text{cat} \in \mathcal{C} \text{ breaking } \mathcal{H})$$

Hardness assumption $\mathcal{H}$ cannot be broken by any $\text{cat} \in \mathcal{C}$

Examples: Discrete-Log, IND-CPA, ...

Involve programming: build cat using cat as sub-procedure.

The Squirrel Prover: Theoretical Foundations

Theoretical foundations: the **CCSA** logic

$$(\text{🐿} \mid \mathcal{P}) \models \Phi$$

1. Modeling

- **Language:** pure λ -calculus
Execution model: encode $(\text{🐿} \mid \mathcal{P})$ interactions
- **FO formulas** for **asymptotic cryptography**
 - **Reachability:** $[\phi_{\mathcal{P}}]$
 - **Indistinguishability:** $\vec{u}_{\mathcal{P}} \sim_c \vec{u}_{\mathcal{P}'}$



2. Reasoning

- **Reasoning rules** valid w.r.t. classical attackers.
Reduction-based rules: $(\exists \text{🐿} \in \mathcal{C}. \dots) \Rightarrow (\exists \text{🐿} \in \mathcal{C}. \dots)$

Building a PQC Verification Framework

$$(\text{cat} \mid \mathcal{P}) \models \Phi$$

Roadmap to adapt a CAC framework to PQC.

- **Modeling:** capture quantum computations and adversaries
- **Reasoning:** capture PQ cryptographic arguments

Building a PQC Verification Framework

$$(\text{cat} \mid \mathcal{P}) \models \Phi$$

Roadmap to adapt a CAC framework to PQC.

- **Modeling:** capture quantum computations and adversaries
- **Reasoning:** capture PQ cryptographic arguments

Challenge (Reasoning)

$$(\exists \text{cat} \in \mathcal{C}. \text{breaking } \mathcal{P}) \Rightarrow (\exists \text{red cat} \in \mathcal{C}. \text{breaking } \mathcal{H})$$

Reductionistic arguments must be adapted:

- Exclude **insecure assumptions**, e.g. Discrete-Log, DDH.
- **No-cloning theorem:** ensure that  is a PQTM

Contributions

- **Execution model** for PQC in Squirrel
- **Faithful logic** for PQC
- **Adapt Squirrel proof systems**
 - Support latest features, e.g. **smt**, **crypto**
- **Implementation**
- **Validation** through **case-studies**
 - Hybrid KEM Combiners
 - Hybrid Key-Exchanges

An PQ Execution Model

Goal: encode $(\text{🐿} \mid \mathcal{P})$ interactions as Squirrel terms

- **Existing encoding** for classical 🐿 **unsuitable**
Require state re-computations $\Rightarrow$ violates no-cloning
- We need a **new execution model for PQC**

Execution Model: Squirrel Primer

Squirrel Language

Pure λ -calculus with **early-sampled randomness**

Execution Model: Squirrel Primer

Squirrel Language

Pure λ -calculus with **early-sampled randomness**

Pure language

Functional encoding with **explicit state**:

 stateful

$\Rightarrow$ **att** stateless

$\text{out} \leftarrow \text{cat}(\text{in}); p$

$\Rightarrow$

$\text{let } (\text{out}, \text{st}') = \text{att}(\text{in}, \text{st}) \text{ in } \tilde{p}$

Execution Model: Squirrel Primer

Squirrel Language

Pure λ -calculus with **early-sampled randomness**

Pure language

Functional encoding with **explicit state**:

 stateful

$\Rightarrow$ **att** stateless

$\text{out} \leftarrow \text{cat}(\text{in}); p$

$\Rightarrow$

$\text{let } (\text{out}, \text{st}') = \text{att}(\text{in}, \text{st}) \text{ in } \tilde{p}$

Early-sampled randomness

Names = arrays of pre-sampled i.i.d. randomness

$n : \text{timestamp} \rightarrow \{0, 1\}^\eta$

$x \xleftarrow{\$} \{0, 1\}^\eta;$

$y \xleftarrow{\$} \{0, 1\}^\eta; p$

$\Rightarrow$

$\text{let } x = n(t) \quad \text{in}$

$\text{let } y = n(t+1) \text{ in } \tilde{p}$

Classical probabilistic machine: bit-string distribution

$$c_{\text{p}}(\text{in}) = \frac{1}{2} \text{"head"} + \frac{1}{2} \text{"tail"}$$

Execution Model: QC Primer

Classical probabilistic machine: bit-string distribution

$$c_{\text{p}}(\text{in}) = \frac{1}{2} \text{"head"} + \frac{1}{2} \text{"tail"}$$

Quantum machine: bit-string superposition

$$c_{\text{q}}(\text{in}) = \frac{1}{\sqrt{2}} |\text{"head"}\rangle - \frac{1}{\sqrt{2}} |\text{"tail"}\rangle$$

Execution Model: QC Primer

Classical probabilistic machine: bit-string distribution

$$c_p(\text{in}) = \frac{1}{2} \text{"head"} + \frac{1}{2} \text{"tail"}$$

Quantum machine: bit-string superposition

$$c_q(\text{in}) = \frac{1}{\sqrt{2}} |\text{"head"}\rangle - \frac{1}{\sqrt{2}} |\text{"tail"}\rangle$$

Measurement: bit-string superposition $\hookrightarrow$ bit-string distribution

$$\frac{1}{\sqrt{2}} |\text{"head"}\rangle - \frac{1}{\sqrt{2}} |\text{"tail"}\rangle \xrightarrow{\text{measure}} \frac{1}{2} \text{"head"} + \frac{1}{2} \text{"tail"}$$

Partial measurement (first N bits):

$$\mathcal{U}_q(\text{in}) \xrightarrow[\text{partial measure (N bits)}]{} \text{Distr}(\{0, 1\}^N \times \mathcal{H}_{\{0,1\}^*})$$

Partial measurement (first N bits):

$$\mathcal{U}_q(\text{in}) \xrightarrow[\text{partial measure (N bits)}]{} \text{Distr}(\{0, 1\}^N \times \mathcal{H}_{\{0,1\}^*})$$

Quantum adversary:

$\{0, 1\}^N$: classical output

$\mathcal{H}_{\{0,1\}^*}$: quantum state

Execution Model: Calling the Quantum Adversary

$$\mathfrak{A}_q(\text{in}) \xrightarrow{\text{partial measure}} \text{Distr}(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*})$$

Modeling a PQTM $\mathfrak{A}_q$ in Squirrel

- Stateless attacker **att** with explicit state **st**
- Pre-sampled randomness for measures:

qrnd : timestamp $\rightarrow$ grand

Encoding of $\mathfrak{A}_q(\text{in})$ for t -th call:

let (out, **st'**) = **att**(**qrnd** t , in, **st**) in ...

Execution Model: Protocol Interaction (Simplified)

Chaining 2 calls

```
in  ← q(out);  
out ←  $\mathcal{P}(\text{in})$ ;  
in  ← q(out);  
...
```

$\Rightarrow$

```
let (in, st) = att(qrnd t, out, st) in  
let out     =  $\tilde{\mathcal{P}}(\text{in})$  in  
let (in, st) = att(qrnd (t + 1), out, st) in  
...
```

Execution Model: Protocol Interaction (Simplified)

Chaining 2 calls

```
in  ← q(out);  
out ←  $\mathcal{P}(\text{in});$   
in  ← q(out);  
...
```

$\Rightarrow$

```
let (in, st) = att(qrnd t, out, st) in  
let out     =  $\tilde{\mathcal{P}}(\text{in})$  in  
let (in, st) = att(qrnd (t + 1), out, st) in  
...
```

Chaining many calls: use **recursive** definitions

```
in (t + 1) = att(qrnd t, out t, st t)#1  
st (t + 1) = att(qrnd t, out t, st t)#2  
out t     =  $\tilde{\mathcal{P}}(\text{in } t)$   
frame t   =  $\langle \text{out init}, \dots, \text{out } t \rangle$ 
```

Execution Model: Conclusion

Key ideas

- **Explicit state** (st t)
- **Measurement randomness** pre-sampled in **qrnd**
 - **qrnd** $\neq$ *program randomness*
 - capture a **physical phenomenon**

Execution Model: Conclusion

Key ideas

- **Explicit state** (st t)
- **Measurement randomness** pre-sampled in **qrnd**
 - **qrnd** $\neq$ *program randomness*
 - capture a **physical phenomenon**

Careful: terms $\neq$ QTM

- Quantum values duplication

$(\mathbf{att}(\mathbf{n}, 0), \mathbf{att}(\mathbf{n}, 0))$

- Weirder, quantum randomness re-use

$(\mathbf{att}(\mathbf{n}, 0), \mathbf{att}(\mathbf{n}, 1))$

Still, all terms have a **well-defined semantics**.

A Faithful Logic for QC

A Faithful Logic for PQC

Cryptographic predicates

■ Reachability $[\phi]$

e.g. $[\text{cat}^*(\text{frame } t) \neq k]$

$$\Pr(\text{not } \llbracket \phi \rrbracket) \leq \epsilon_{\text{negl}}$$

■ Indistinguishability $u \sim v$

e.g. $\text{frame}_{\mathcal{P}} t \sim \text{frame}_{\mathcal{P}'} t$

Classical $u \sim_c v$

$$\forall \text{cat}^* : \text{PPTM}. \quad |\Pr(\text{cat}^*(\llbracket u \rrbracket) = 1) - \Pr(\text{cat}^*(\llbracket v \rrbracket) = 1)| \leq \epsilon_{\text{negl}}$$

A Faithful Logic for PQC

Cryptographic predicates

■ Reachability $[\phi]$

e.g. $[\text{cat}^*(\text{frame } t) \neq k]$

$$\Pr(\text{not } \llbracket \phi \rrbracket) \leq \epsilon_{\text{negl}}$$

■ Indistinguishability $u \sim v$

e.g. $\text{frame}_{\mathcal{P}} t \sim \text{frame}_{\mathcal{P}'} t$

Classical $u \sim_c v$

$$\forall \text{cat}^* : \text{PPTM}. |\Pr(\text{cat}^*(\llbracket u \rrbracket) = 1) - \Pr(\text{cat}^*(\llbracket v \rrbracket) = 1)| \leq \epsilon_{\text{negl}}$$

Quantum $u \sim_q v$

$$\forall \text{cat}^* : \text{PQTM}. |\Pr(\text{cat}^*(\llbracket u \rrbracket) = 1) - \Pr(\text{cat}^*(\llbracket v \rrbracket) = 1)| \leq \epsilon_{\text{negl}}$$

A Faithful Logic for PQC

Early-sampled probabilities

Names: **finite** arrays of pre-sampled randomness.

$$n : S \rightarrow D$$

Finiteness ensures that:

$$\Pr_{\rho} \left(\llbracket u \rrbracket(\rho) \in E \right) \text{ well-defined for any } E$$

Otherwise, must ensure measurability of $(\llbracket u \rrbracket \in E) \Rightarrow$ complex in general

A Faithful Logic for PQC

Early-sampled probabilities

Names: **finite** arrays of pre-sampled randomness.

$$n : S \rightarrow D$$

Finiteness ensures that:

$$\Pr_{\rho} \left(\llbracket u \rrbracket(\rho) \in E \right) \text{ well-defined for any } E$$

Otherwise, must ensure measurability of $(\llbracket u \rrbracket \in E) \Rightarrow$ complex in general

Quantum measurement modeling

- $(\text{qrnd } t)$ as large as we want

- **But** same size for all terms

$\Rightarrow$ **not always enough randomness!**

A Faithful Logic for PQC

Solution

■ Approximation models $\mathbb{M}$:

finite (qrnd t) faithful meas. $\times$ $\Pr(\llbracket u \rrbracket_{\mathbb{M}} \in E) \checkmark$

■ Exact models $\mathbb{M}_e$:

infinite (qrnd t) faithful meas. $\checkmark$ $\Pr(\llbracket u \rrbracket_{\mathbb{M}_e} \in E) ?$

Adequacy Theorem

If u is **well-formed** and if (qrnd t) is **large enough**:

$$D_{\text{stat}}(\llbracket u \rrbracket_{\mathbb{M}}, \llbracket u \rrbracket_{\mathbb{M}_e}) \leq \epsilon_{\text{negl}}$$

(Very roughly, well-formed = PQTM simulatable)

- **Execution model** for PQC in Squirrel ✓
- **Faithful logic** for PQC ✓
- **Adapt Squirrel proof systems**
- **Implementation** + **Case-studies**

Adapting Squirrel Proof System

Adapting the Proof System

Goal: adapt Squirrel reasoning capabilities to PQC

- **Core logical rules:** `rewrite`, `apply`, `smt`, ...
- **Cryptographic rules:**
 - Basic rules, e.g. `fresh`, `fa`, `dup`, ...
 - Automated simplifications: `deduce`
 - Reductions to hardness assumptions: `crypto`

Adapting the Proof System: Core Logical Rules

- We use Squirrel existing semantics.
- $\sim_c$ and $\sim_q$ can **absorb a negligible error**.

⇒ **inherit non-reductionist rules for free.**

$$\begin{array}{ccc} \text{rewrite}_c & & \text{rewrite}_q \\ \frac{u' \sim_c v \quad [u = u']}{u \sim_c v} & \Rightarrow & \frac{u' \sim_q v \quad [u = u']}{u \sim_q v} \end{array} \qquad \frac{\text{smt} \quad \vdash_{\text{smt}} \phi}{[\phi]}$$

Adapting the Proof System

- **Core logical rules:** `rewrite` ✓, `apply` ✓, `smt` ✓, ...
- **Cryptographic rules:**
 - Basic rules, e.g. `fresh` ✓, `fa`, `dup`, ...
 - Automated simplifications: `deduce`
 - Reductions to hardness assumptions: `crypto`

Difficulty

Remaining rules are **reduction-based**.

Reductionist Rules: Basic Rules

Classical rules

$$\text{dup}_c \frac{u \sim_c v}{u, u \sim_c v, v}$$

$$\text{fa}_c \frac{u \sim_c v \quad f \in \text{Lib}}{f(u) \sim_c f(v)}$$

Reductionist Rules: Basic Rules

Classical rules

$$\text{dup}_c \frac{u \sim_c v}{u, u \sim_c v, v}$$

$$\text{fa}_c \frac{u \sim_c v \quad f \in \text{Lib}}{f(u) \sim_c f(v)}$$

Issues

$$\frac{u_q \sim_q v_q}{u_q, u_q \sim_q v_q, v_q}$$

cloning quantum value **X**

$$\frac{\text{att}(r, u) \sim_q v}{\text{att}(r, \text{att}(r, u)) \sim_q \text{att}(r, v)}$$

quantum randomness **r** re-used **X**

Quantum rules

$$\text{dup}_q \frac{u_c \sim_c v_c}{u_c, u_c \sim_c v_c, v_c}$$

$$\text{fa}_q \frac{u \sim_q v \quad \phi_{\text{fresh}}^r(u, v)}{\text{att}(r, u) \sim_q \text{att}(r, v)}$$

Adapting the Proof System

- **Core logical rules:** `rewrite` ✓, `apply` ✓, `smt` ✓, ...
- **Cryptographic rules:**
 - Basic rules, e.g. `fresh` ✓, `fa` ✓, `dup` ✓, ...
 - Automated simplifications: `deduce`
 - Reductions to hardness assumptions: `crypto`

Reductionist Rules: Bi-Deduction

Tactic (deduce): automated simplification by simple reduction [CSF'22]

$\sim_c$ preserved by public **deterministic** computation

■ **predicate:** $u \triangleright_c v$

■ **proof system:** $\vdash_c$

Reductionist Rules: Bi-Deduction

Tactic (deduce): automated simplification by simple reduction [CSF'22]

$\sim_c$ preserved by public **deterministic** computation

■ **predicate:** $u \triangleright_c v$

■ **proof system:** $\vdash_c$

Quantum adaptation

■ **predicate** $\triangleright_q$: **deterministic** $\Rightarrow$ **error-free**

$$f(\text{in}) \xrightarrow{\text{partial measure}} \text{Dirac}(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}) \approx \{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}$$

■ **proof system:** $\vdash_q = \vdash_c + \text{linearity conditions}$

Reductionist Rules: Cryptographic Bi-Deduction

Tactic (crypto): automated cryptographic reductions to $\mathcal{G}$ [CCS'24]

- **Predicate:** $\dots \vdash \mathbf{u} \triangleright_c^{\mathcal{G}} \mathbf{v}$
- **Complex semantics:** probabilistic reductions + $\mathcal{G}$ stateful ($\dots$)
- **Proof system** for $\triangleright_c^{\mathcal{G}}$
- **Automatic proof-search**, including induction

Reductionist Rules: Cryptographic Bi-Deduction

Challenges

- Probabilistic reductions $\Rightarrow$ quantum error-free **insufficient**
- Generalization to PQTM **complex**:
 $\Rightarrow$ change semantics + rules + proof-search **X**

Key Idea

- **Encapsulate quantum manipulation** in the game
- **Safe quantum API** $\mathcal{Q}$: force **linear usage** of quantum state



Case-Studies

Case studies in our post-quantum Squirrel

- **KEM combiners (CPA/CCA):**

XOR, XOR-then-MAC, Dual-PRF, Nested Dual-PRF

- **Hybrid KEM-based KE protocols (strong secrecy):**

BCGNP [S&P'22], C_{Sigma}

- Use latest features: **crypto**, **smt**

Proof strategy

- First proof in classical setting ($\approx$ pers. **months**)

- Then, adapted to post-quantum ($\approx$ pers. **days**)

Conclusion

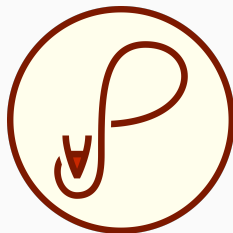
Conclusion

- Execution model for PQC
- A faithful logic for PQC
Adequacy theorem
- Adapt Squirrel proof systems
smt, crypto, ...
- Implementation + Case-studies
Hybrid KEM Combiners and KE



Conclusion

- Execution model for PQC
- A faithful logic for PQC
Adequacy theorem
- Adapt Squirrel proof systems
smt, crypto, ...
- Implementation + Case-studies
Hybrid KEM Combiners and KE



Thank you for your attention

Reductionist Rules: Bi-Deduction

Advantages of PQTM_E (error-free quantum machines)

- No measurement randomness
⇒ no probabilistic dependencies ✓
- Support basic manipulations of quantum values
Example: swapping $(c, q) \triangleright (q, c)$
- For more complex manipulations:
automatic **deduce** + manual **fa** for **att**(.)

Reductionist Rules: Bi-Deduction

Proof system

$$\triangleright_q \approx \triangleright_c + \begin{array}{l} \text{linear usage} \\ \text{of quantum values} \end{array}$$

Example: transitivity

$$\frac{u \triangleright_c w \quad u, w \triangleright_c v}{u \triangleright_c v} \Rightarrow \frac{c, q_1 \triangleright_q w \quad c, q_2, w \triangleright_q v}{c, q_1, q_2 \triangleright_q v}$$

- **Classical** value **c** can be re-used
- **Quantum** value **q₁, q₂** used **linearly**

Reductionist Rules: Cryptographic Bi-Deduction

Safe Quantum API

```
game  $\mathcal{Q} = \{$   
  (* quantum state *)  
  var state :  $\mathcal{H}_{\text{message}}$  = ...;  
  
  (* classical state, next protocol input *)  
  var input : message = ...;  
  
  (* update state using att *)  
  oracle step (out) = {  
    r  $\xleftarrow{\$}$  qrand;  
    (input, state) = att(r, (state, out));  
  }  
  
  (* retrieve last attacker input *)  
  oracle get_input () = { return input; }  
}
```

Reductionist Rules: Cryptographic Bi-Deduction

$$\exists A : \text{PPTM} . A^{\mathcal{Q} \cdot \mathcal{G}} = u \quad \Rightarrow \quad \exists B : \text{PQTM} . B^{\mathcal{G}} = u$$

$$B^{\mathcal{G}} = (A^{\mathcal{Q}})^{\mathcal{G}} = A^{\mathcal{Q} \cdot \mathcal{G}}$$

- **Semantics** of $\triangleright_c^{\mathcal{G}}$ unchanged (except minor change for $\mathcal{G}$ quantum)
- **Proof system** unchanged
- Specialized **induction rule** for the quantum execution model

$$\frac{x \vdash \text{in } x \triangleright_c^{\mathcal{G}} \text{ out } x}{t \triangleright_c^{\mathcal{G} \cdot \mathcal{Q}} \text{ frame } t}$$

No manipulation of **state** by S .

- **Implementation**: re-use most machinery