Robust Logical Foundations for Mechanizing Post-Quantum Cryptography in Squirrel

CCS 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

November 16, 2026, The Hague, Netherlands



Context

Provable Security

Cryptographic protocols are **critical**. https://



Computer-Aided Cryptography (CAC) provides a high level of confidence
provable security + **mechanization**

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

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

Classical attacker model: $\mathcal{C} = \mathrm{PPTM}$

Quantum Computers

The **attacker model** should cover all realistic attack scenario

Quantum Computers (QC): PPTM $\Rightarrow$ P**Q**TM

Post-Quantum Cryptography (PQC)

Secure cryptography against quantum adversaries.

$$\forall \mathcal{A} \in \mathsf{PQTM}.\ \Pr(\mathcal{A} \text{ breaks } \mathcal{P}) \leq \epsilon$$

[**quantum** adversary
classical protocol]

PQC Frameworks: CryptoVerif, EasyPQC, **PQ-Squirrel**, qrhl-tool

Verification frameworks for PQC are still at an early stage.

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

- **Expressivity:** QROM not supported
- **Maintainability:**
  - **Unusual** stateful semantics
  - **Distance** with implementation
  - **Lacks** latest improvements
    no smt, no crypto (auto. reductions)

PQ-Squirrel

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

- **Expressivity:** QROM not supported
- **Maintainability:**
  - **Unusual** stateful semantics
  - **Distance** with implementation
  - **Lacks** latest improvements
    no smt, no crypto (auto. reductions)

Goal

Design a **robust** PQ extension of Squirrel

Cryptographic Reductions

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

Cryptographic reductions: proof technique for provable security

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

Involve programming: build 😈 using 😼 as sub-procedure.

The Squirrel Prover: Theoretical Foundations

Theoretical foundations: a **CCSA logic**

$(\mathcal{A} \mid \mathcal{P}) \quad \models \quad \Phi$



1. Modeling

- **Language:** pure λ-calculus
  Execution model
  encodes $(\mathcal{A} \mid \mathcal{P})$ interactions
- **Logic** for **asymptotic cryptography**
  - **Reachability**: $[\phi_{\mathcal{P}}]$
  - **Indistinguishability:** $\mathcal{P} \sim_c \mathcal{Q}$

2. Reasoning

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

Building a PQC Verification Framework

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

Roadmap to adapt a CAC framework to PQC.

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

Building a PQC Verification Framework

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

Roadmap to adapt a CAC framework to PQC.

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

Challenge (Modeling)

Squirrel semantics:
early-sampled + discrete

$\Rightarrow$ Faithfully **modeling quantum measurements** is tricky.

Challenge (Reasoning)

$$(\exists \text{🐱} \in \mathcal{C}. \ldots) \Rightarrow (\exists \text{😈} \in \mathcal{C}. \ldots)$$

No-cloning theorem: $\mathsf{q} \not\Rightarrow (\mathsf{q}, \mathsf{q})$
Reductionistic arguments must be adapted to ensure that 😈 is a PQTMs

Contributions

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

A Faithful Logic for PQC in Squirrel

Execution Model

Goal: encode (| $\mathcal{P}$) 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**

$\ldots$
out $\leftarrow \mathcal{P}$(in)
in $\leftarrow$ (out)
$\ldots$

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**

```
...
out ← 𝒫(in)
in ← 🐱(out)
...
```

Interaction with the adversary 🐱

$$\text{🐱}(\text{out}) \xleftrightarrow[\substack{\text{computation}\\+\\\text{quantum}\\\text{measurement}}]{} \text{Distr}\Big(\underbrace{\{0,1\}^*}_{\substack{\text{protocol}\\\text{input}}} \times \underbrace{\mathcal{H}_{\{0,1\}^*}}_{\substack{\text{updated}\\\text{state}}} \Big)$$

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 $\rightsquigarrow$ functional encoding with **explicit state**

 stateful $\Rightarrow$ **att** stateless

$\text{in} \leftarrow$ (out) $\Rightarrow$ **let** $(\text{in}, \text{st}') = \textbf{att}(\text{out}, \text{st})$

Execution Model: Squirrel Primer

Squirrel Language

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

Pure language $\rightsquigarrow$ functional encoding with **explicit state**

stateful $\Rightarrow$ **att** stateless

$\text{in} \leftarrow$ (out) $\Rightarrow$ **let** $(\text{in}, \text{st}') = \textbf{att}(\text{out}, \text{st})$

Early-sampled randomness $\rightsquigarrow$ use names (arrays of **pre-sampled** i.i.d. randomness)

$$\textbf{let}\ (\text{out}, \text{st}') = \textbf{att}(\text{qrnd}\ t, \text{in}, \text{st})$$

Faithfulness of the Semantics

Difficulty

- Squirrel's semantics is **discrete** to avoid **measurability issues**. Thus, **names** are **finite**.
- (qrnd t) as large as we want **but** same size for all terms

$\Rightarrow$ **not always enough randomness for att**(qrnd t, in, st)**!**

Faithfulness of the Semantics

Difficulty

- Squirrel's semantics is **discrete** to avoid **measurability issues**. Thus, **names** are **finite**.
- (qrnd t) as large as we want **but** same size for all terms

$\Rightarrow$ **not always enough randomness for att**(qrnd t, in, st)**!**

Adequacy Theorem

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

$$\mathsf{D}_{\mathsf{stat}}(\llbracket u \rrbracket_{\mathbb{M}}, \llbracket u \rrbracket_{\mathbb{M}_e}) \leq \epsilon_{\mathsf{negl}}$$

(well-formed $\approx$ PQTM simulatable)

Outline

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

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**.

$\Rightarrow$ **inherit non-reductionist rules for free.**

$$\frac{u_1 \sim_c v \qquad [u_0 = u_1]}{u_0 \sim_c v}\ \text{rewrite}_c \quad \Rightarrow \quad \frac{u_1 \sim_q v \qquad [u_0 = u_1]}{u_0 \sim_q v}\ \text{rewrite}_q \qquad \frac{\vdash_{\text{smt}} \phi}{[\phi]}\ \text{smt}$$

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**.

Adapt the rules: **restrictions**, **side-conditions**, **linearity**, ...

Reductionist Rules: Basic Rules

Basic rules (dup, fa)

$$\frac{u \sim_{\mathsf{c}} v}{u, u \sim_{\mathsf{c}} v, v} \qquad \frac{s \sim_{\mathsf{c}} t \qquad f \in \mathsf{Lib}}{f(s) \sim_{\mathsf{c}} f(t)}$$

$\Rightarrow$ **restrictions** to classical terms + $\phi^{\mathsf{r}}_{\mathsf{fresh}}$ **side-conditions** for $\mathbf{att}(\mathsf{r}, \cdot)$

Bideduction (deduce)

automated **deterministic** reduction

$$\mathbf{u} \rhd_{\mathsf{c}} \mathbf{v} \;=\; \exists \mathcal{S} \in \mathsf{PTM},\ \mathcal{S}(\mathbf{u}) = \mathbf{v}$$

proof system $\vdash_{\mathsf{c}}$

$\Rightarrow$ $\rhd_{\mathsf{q}}$: ~~**deterministic**~~ $\Rightarrow$ **error-free**

$\vdash_{\mathsf{q}} \;=\; \vdash_{\mathsf{c}}$ + **linearity conditions**

Crypto. bideduction (crypto)

automated reductions to game $\mathcal{H}$

$$\mathbf{u} \rhd_{\mathsf{c}}^{\mathcal{H}} \mathbf{v} \;=\; \exists \mathcal{S} \in \mathbf{P}\mathsf{PTM},\ \mathcal{S}^{\mathcal{H}}(\mathbf{u}) = \mathbf{v}$$

proof system + **proof-search**

$\Rightarrow$ **safe quantum API** $\mathcal{Q}$

force **linear usage** of quantum state

$$\rhd_{\mathsf{q}}^{\mathcal{H}} \quad \Rightarrow \quad \rhd_{\mathsf{c}}^{\mathcal{H} \cdot \mathcal{Q}}$$

reuse most rules and proof-search

Outline

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

Implementation and Case Studies

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

- **Faithful logic** for PQC in Squirrel
  execution model, **adequacy theorem**
- **Adapt Squirrel proof systems**
  smt, crypto, ...
- **Implementation** + **Case-studies**
  hybrid KEM Combiners and KE



Conclusion

- **Faithful logic** for PQC in Squirrel
  execution model, **adequacy theorem**
- **Adapt Squirrel proof systems**
  smt, crypto, ...
- **Implementation** + **Case-studies**
  hybrid KEM Combiners and KE



Thank you for your attention

Faithfulness of the Semantics

Difficulty

Squirrel's semantics is **discrete**. **Finiteness** of **names** ensures that

$$\Pr\left([\![u]\!] \in \mathrm{E}\right) \text{ well-defined} \qquad \text{(for any } E\text{)}$$

(qrnd t) as large as we want **but** same size for all terms

$\Rightarrow$ **not always enough randomness for att**(qrnd $t, \mathsf{in}, \mathsf{st}$)**!**

Faithfulness of the Semantics

Difficulty

Squirrel's semantics is **discrete**. **Finiteness** of **names** ensures that

$$\Pr\left([\![u]\!] \in \mathsf{E}\right) \text{ well-defined} \qquad \text{(for any } E\text{)}$$

(qrnd t) as large as we want **but** same size for all terms

$\Rightarrow$ **not always enough randomness for att**(qrnd t, in, st)**!**

Approximation model $\mathbb{M}$

finite (qrnd t) faithful meas. ✗

$\Pr([\![u]\!]_{\mathbb{M}} \in \mathsf{E})$ ✓

Exact model $\mathbb{M}_{\mathsf{e}}$

infinite (qrnd t) faithful meas. ✓

$\Pr([\![u]\!]_{\mathbb{M}_{\mathsf{e}}} \in \mathsf{E})$?

Faithfulness of the Semantics

Difficulty

Squirrel's semantics is **discrete**. **Finiteness** of **names** ensures that

$$\Pr\left([\![u]\!] \in \mathsf{E}\right) \text{ well-defined} \qquad \text{(for any } E\text{)}$$

(qrnd t) as large as we want **but** same size for all terms
$\Rightarrow$ **not always enough randomness for att**(qrnd t, in, st)**!**

Approximation model $\mathbb{M}$

finite (qrnd t) faithful meas. ✗

$\Pr([\![u]\!]_{\mathbb{M}} \in \mathsf{E})$ ✓

Exact model $\mathbb{M}_{\mathsf{e}}$

infinite (qrnd t) faithful meas. ✓

$\Pr([\![u]\!]_{\mathbb{M}_{\mathsf{e}}} \in \mathsf{E})$?

Adequacy Theorem

u is **well-formed** + (qrnd t) **large enough**

$$\mathsf{D}_{\mathsf{stat}}\left([\![u]\!]_{\mathbb{M}}, [\![u]\!]_{\mathbb{M}_{\mathsf{e}}}\right) \leq \epsilon_{\mathsf{negl}}$$

(well-formed $\approx$ PQTM simulatable)

Reductionist Rules: Basic Rules

Classical rules

$$\mathsf{dup_c}\ \frac{u \sim_{\mathsf{c}} v}{u, u \sim_{\mathsf{c}} v, v} \qquad \mathsf{fa_c}\ \frac{u \sim_{\mathsf{c}} v \qquad f \in \mathrm{Lib}}{f(u) \sim_{\mathsf{c}} f(v)}$$

Issues

$$\frac{u_q \sim_{\mathsf{q}} v_q}{u_q, u_q \sim_{\mathsf{q}} v_q, v_q}$$

cloning quantum value ✗

$$\frac{\mathbf{att}(\mathsf{r}, u) \sim_{\mathsf{q}} v}{\mathbf{att}(\mathsf{r}, \mathbf{att}(\mathsf{r}, u)) \sim_{\mathsf{q}} \mathbf{att}(\mathsf{r}, v)}$$

quantum randomness r re-used ✗

Quantum rules

$$\mathsf{dup_q}\ \frac{u_c \sim_{\mathsf{c}} v_c}{u_c, u_c \sim_{\mathsf{c}} v_c, v_c} \qquad \mathsf{fa_q}\ \frac{u \sim_{\mathsf{q}} v \qquad \phi^{\mathsf{r}}_{\mathsf{fresh}}(u, v)}{\mathbf{att}(\mathsf{r}, u) \sim_{\mathsf{q}} \mathbf{att}(\mathsf{r}, v)}$$

Reductionist Rules: Bi-Deduction

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

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

- **predicate:** $\mathbf{u} \rhd_c \mathbf{v}$
- **proof system:** $\vdash_c$

Quantum adaptation

- **predicate** $\rhd_q$: ~~**deterministic**~~ $\Rightarrow$ **error-free**

$$f(\mathsf{in}) \xhookrightarrow[\text{partial measure}]{} \mathrm{Dirac}\big(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}\big) \approx \{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}$$

- **proof system:** $\vdash_q \;=\; \vdash_c +$ **linearity conditions**

Reductionist Rules: Bi-Deduction

Advantages of $\mathbf{PQTM_E}$ (error-free quantum machines)

- No measurement randomness
  $\Rightarrow$ no probabilistic dependencies ✓
- Support basic manipulations of quantum values
  Example: swapping $(c, q) \rhd (q, c)$
- For more complex manipulations:
  automatic deduce + manual fa for $\mathbf{att}(\cdot)$

Reductionist Rules: Bi-Deduction

Proof system

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

Example: transitivity

$$\frac{\mathbf{u} \triangleright_c \mathbf{w} \qquad \mathbf{u}, \mathbf{w} \triangleright_c \mathbf{v}}{\mathbf{u} \triangleright_c \mathbf{v}} \quad \Rightarrow \quad \frac{\mathbf{c}, \mathbf{q}_1 \triangleright_q \mathbf{w} \qquad \mathbf{c}, \mathbf{q}_2, \mathbf{w} \triangleright_q \mathbf{v}}{\mathbf{c}, \mathbf{q}_1, \mathbf{q}_2 \triangleright_q \mathbf{v}}$$

- **Classical** value $\mathbf{c}$ can be re-used
- **Quantum** value $\mathbf{q}_1$, $\mathbf{q}_2$ used **linearly**

Reductionist Rules: Cryptographic Bi-Deduction

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

- **Predicate:** $\cdots \vdash \mathbf{u} \rhd_{\mathsf{c}}^{\mathcal{G}} \mathbf{v}$
- **Complex semantics**: probabilistic reductions + $\mathcal{G}$ stateful ($\cdots$)
- **Proof system** for $\rhd_{\mathsf{c}}^{\mathcal{G}}$
- **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 ✗

Key Idea

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

$$\triangleright \begin{matrix} \mathcal{G} \\ \mathsf{q} \end{matrix} \qquad \Rightarrow \qquad \triangleright \begin{matrix} \mathcal{G} \cdot \mathcal{Q} \\ \mathsf{c} \end{matrix}$$

Reductionist Rules: Cryptographic Bi-Deduction

Safe Quantum API

```
game 𝒬 = {
  (* quantum state *)
  var state : ℋ_message = ...;

  (* classical state, next protocol input *)
  var input : message = ...;

  (* update state using att *)
  oracle step (out) = {
    r ←$ qrand;
    (input,state) = att(r, (state,out));
  }

  (* retrieve last attacker input *)
  oracle get_input () = { return input; }
}
```

Reductionist Rules: Cryptographic Bi-Deduction

$$\exists\, \mathsf{A} : \mathsf{PPTM}\,.\, \mathsf{A}^{\mathcal{Q}\cdot\mathcal{G}} = u \qquad \Rightarrow \qquad \exists\, \mathsf{B} : \mathsf{PQTM}\,.\, \mathsf{B}^{\mathcal{G}} = u$$

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

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

$$\frac{x \vdash \mathsf{in}\ x \rhd_{\mathsf{c}}^{\mathcal{G}}\ \mathsf{out}\ x}{t \rhd_{\mathsf{c}}^{\mathcal{G}\cdot\mathcal{Q}}\ \mathsf{frame}\ t}$$

  No manipulation of state by S.
- **Implementation**: re-use most machinery