Robust Logical Foundations for Mechanizing Post-Quantum Cryptography in Squirrel

SVP/PQ-TLS Workshop

David Baelde	Univ Rennes, IRISA, CNRS
Antoine Dallon	AMIAD
Stéphanie Delaune	Univ Rennes, IRISA, CNRS
Charlie Jacomme	Inria Nancy
Adrien Koutsos	Inria Paris

3rd of February 2026, Rennes

Context

Context

Computer-Aided Cryptography (CAC)

- **Cryptographic proofs**: formal proof of security.
- **Mechanization**: high level of confidence.

Example:

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

Standard cryptography: $\mathcal{C} = \text{PPTM}$

- *Polynomial-time*
- *Probabilistic*
- (**classical**) *Turing Machine*

CAC Frameworks: CryptoVerif, Squirrel, EasyCrypt, SSProve

Quantum Computers

- Working quantum computer (QC) may arrive
- QC breaks many existing crypto systems
  Discrete logarithm, Diffie-Hellman, ...

Post-Quantum Cryptography (PQC)

Secure cryptography against quantum adversaries.

- adversary: **quantum**
- protocol: **classical**

PQC $\neq$ quantum cryptography (protocol: quantum).

Context: Post-Quantum Cryptography

PQC effort in progress

- PQ primitives: ML-KEM, ML-DSA
- PQ protocols: Signal (PQXDH, SPQR), iMessage (PQ3)

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

Mechanized cryptographic proofs of PQ security.

$$\forall \mathcal{A} \in \mathcal{C}.\ \Pr(\mathcal{A} \text{ breaks } \mathcal{P}) \leq \epsilon \qquad (\mathcal{C} = \mathrm{P}\mathbf{Q}\mathrm{TM})$$

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

$\neq$ tools $\Rightarrow$ $\neq$ strengths

Context: PQ-Squirrel

Limitations of PQ-Squirrel

- **Expressivity:**

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

  $\Rightarrow$ quantum values not represented (e.g. no QROM)
- **Unusual semantics:**
  - Maintainability (implem)
  - Lacks latest improvements (theory, implem), e.g. crypto, smt

Goal: improved PQ version of Squirrel

The Squirrel Prover: Theoretical Foundations

Theoretical foundations: the **CCSA logic**

1. Modeling

- **Language:** pure λ-calculus
  Execution model: encode $(\mathcal{A}|\mathcal{P})$ 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.
- **Automation** for cryptographic reasoning.

The Squirrel Prover: Implementation

Proof assistant

Users prove goals using tactics.

- **Generic maths**, e.g. apply, rewrite, smt.
- **Crypto**, e.g. trans, deduce, crypto.
  Automated **simulator synthesis** procedures.

Development done in Proof-General.

As in Rocq, EasyCrypt ...

Open-source: `https://squirrel-prover.github.io/`

Context: Building a PQC Verification Framework

Roadmap to adapt a CAC framework to PQC.

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

Challenge (Reasoning)

Reductionistic arguments must be adapted:

- Exclude insecure assumptions, e.g. DDH.
- **No-cloning theorem:** ensure that simulators are PQTMs

Contributions

- New **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

Execution Model

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

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

Execution Model: Squirrel Primer

Pure language

Functional encoding with **explicit state**:

$$\mathcal{A} \text{ stateful} \quad \Rightarrow \quad \textbf{att} \text{ stateless}$$

$$(\text{out} \leftarrow \mathcal{A}(\text{in}); p) \quad \Rightarrow \quad \textbf{let } (\text{out}, \text{st}') = \textbf{att}(\text{in}, \text{st}) \textbf{ in } \tilde{p}$$

Early-sampled randomness

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

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

$$\begin{matrix} \text{x} \xleftarrow{\$} \{0,1\}^\eta; \\ \text{y} \xleftarrow{\$} \{0,1\}^\eta; \ldots \end{matrix} \quad \Rightarrow \quad \begin{matrix} \textbf{let } \text{x} = \text{n } t \quad \textbf{in} \\ \textbf{let } \text{y} = \text{n } (\text{next } t) \textbf{ in } \ldots \end{matrix}$$

Execution Model: QC Primer

Classical (probabilistic) machine $\mathcal{A}_\mathsf{c}$

$$\mathcal{A}_\mathsf{c}(\mathsf{in}) = \sum_{v \in \{0,1\}^*} p_v \cdot v \qquad \sum_v p_v = 1, \ \ \forall v.\, p_v \in \mathbb{R}^+$$

Example: $\frac{1}{2} \cdot \texttt{"pq-tls"} + \frac{1}{2} \cdot \texttt{"svp"}$

Quantum machine $\mathcal{A}_\mathsf{q}$

$$\mathcal{A}_\mathsf{q}(\mathsf{in}) = \sum_{v \in \{0,1\}^*} q_v \cdot |v\rangle \qquad \sum_v |q_v|^2 = 1, \ \ \forall v.\, q_v \in \mathbb{C}$$

Example: $\frac{1}{\sqrt{2}} \cdot |\texttt{"pq-tls"}\rangle - \frac{1}{\sqrt{2}} \cdot |\texttt{"svp"}\rangle$

Execution Model: QC Primer

$$\mathcal{A}_{\mathsf{q}}(\mathsf{in}) = \sum_{v \in \{0,1\}^*} q_v \cdot |v\rangle \qquad\qquad \sum_v |q_v|^2 = 1$$

Measurement yield v with proba. $|q_v|^2$

Partial measurement (first N bits):

$$\mathcal{A}_{\mathsf{q}}(\mathsf{in}) \xhookrightarrow[\substack{\text{partial measure}\\ \text{(N bits)}}]{} \mathsf{Distr}\left(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}\right)$$

Execution Model: Calling the Quantum Adversary

$$\mathcal{A}_{\mathsf{q}}(\mathsf{in}) \xhookrightarrow[\text{partial measure}]{} \mathsf{Distr}\big(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}\big)$$

Modeling a PQTM $\mathcal{A}_{\mathsf{q}}$ in Squirrel

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

$$\mathsf{qrnd} : \mathsf{timestamp} \to \mathsf{qrand}$$

Encoding of $\mathcal{A}_{\mathsf{q}}(\mathsf{in})$ for t-th call:

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

Execution Model: Protocol Interaction (Simplified)

Chaining 2 calls

$$
\begin{array}{lll}
\mathsf{in} & \leftarrow & \mathcal{A}_{\mathsf{q}}(\mathsf{out}); \\
\mathsf{out} & \leftarrow & \mathcal{P}(\mathsf{in}); \\
\mathsf{in} & \leftarrow & \mathcal{A}_{\mathsf{q}}(\mathsf{out}); \\
\ldots & &
\end{array}
\quad \Rightarrow \quad
\begin{array}{llll}
\textbf{let}\ (\mathsf{in}, \mathsf{st}) & = & \textbf{att}(\mathsf{qrnd}\ t, \qquad (\mathsf{out}, \mathsf{st})) & \textbf{in} \\
\textbf{let}\ \mathsf{out} & = & \tilde{\mathcal{P}}(\mathsf{in}) & \textbf{in} \\
\textbf{let}\ (\mathsf{in}, \mathsf{st}) & = & \textbf{att}(\mathsf{qrnd}\ (\mathsf{next}\ t),(\mathsf{out}, \mathsf{st})) & \textbf{in} \\
\ldots & & &
\end{array}
$$

Chaining many calls: use **recursive** definitions

$$
\begin{array}{rcl}
\mathsf{in}\ (\mathsf{next}\ t) & = & \textbf{att}(\mathsf{qrnd}\ t, (\mathsf{out}\ t, \mathsf{st}\ t))\#1 \\
\mathsf{st}\ (\mathsf{next}\ t) & = & \textbf{att}(\mathsf{qrnd}\ t, (\mathsf{out}\ t, \mathsf{st}\ t))\#2 \\
\mathsf{out}\ t & = & \tilde{\mathcal{P}}(\mathsf{in}\ t) \\
\mathsf{frame}\ t & = & \langle \mathsf{out}\ \mathsf{init}, \ldots, \mathsf{out}\ t \rangle
\end{array}
$$

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

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

- Weirder, quantum randomness re-use

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

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

A Faithful Logic for QC

A Faithful Logic for PQC

Cryptographic predicates

- **Reachability $[\phi]$, no changes**

$$\Pr(\textbf{not } [\![\phi]\!]) \leq \epsilon_{\text{negl}}$$

- **Indistinguishability**

Classical $u \sim_c v$

$$\forall \mathcal{A} : \text{PPTM}. \quad |\Pr(\mathcal{A}([\![u]\!]) = 1) - \Pr(\mathcal{A}([\![v]\!]) = 1)| \leq \epsilon_{\text{negl}}$$

Quantum $u \sim_q v$

$$\forall \mathcal{A} : \text{PHTM}. \quad |\Pr(\mathcal{A}([\![u]\!]) = 1) - \Pr(\mathcal{A}([\![v]\!]) = 1)| \leq \epsilon_{\text{negl}}$$

A Faithful Logic for PQC

Hybrid machine $\mathcal{A}$: PHTM

$$\mathcal{A}(\mathsf{in}) = \mathsf{fold}(\mathcal{A}_\mathsf{c}, \mathcal{A}_\mathsf{q}, S_\$, \mathsf{in})$$

- $\mathcal{A}_\mathsf{c}$: PPTM, $\mathcal{A}_\mathsf{q}$: PQTM
- $S_\$ = \{r_1, \ldots, r_N\}$ sampled in qrand^N
- Full computation in **polynomial-time**

Advantages:

- Simplify soundness of reasoning rules
- More **expressive**:
  - $\mathcal{A}_\mathsf{c}$ can have **classical oracles**
  - $\mathcal{A}_\mathsf{q}$ can have **quantum oracles**

A Faithful Logic for PQC

Early-sampled probabilities

Finite arrays of pre-sampled randomness ρ. Ensures that:

$$\Pr_{\rho}(\llbracket u \rrbracket(\rho) \in \mathsf{E}) \text{ well-defined}$$

Quantum measurement modeling

- ($\mathsf{qrnd}\ t$) as large as we want
- **But** same size for all terms

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

A Faithful Logic for PQC

Solution

- **Approximation models** $\mathbb{M}$:

$$\text{finite } (\text{qrnd } t) \qquad \Pr([\![u]\!]_{\mathbb{M}} \in \mathsf{E}) \checkmark$$

- **Exact models** $\mathbb{M}_\mathsf{e}$:

$$\text{infinite } (\text{qrnd } t) \qquad \Pr([\![u]\!]_{\mathbb{M}_\mathsf{e}} \in \mathsf{E}) \ ?$$

Adequacy Theorem

For well-formed terms t:

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

(Very roughly, well-formed = PQTM simulatable)

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. trans, fresh, fa, . . .
  - 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.**

Examples:

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

Adapting the Proof System

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

Difficulty

Remaining rules are **reduction-based**.

Reductionist Rules: Basic Rules

Classical function application:

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

Issue: quantum randomness r re-used

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

Quantum function application:

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

- $\phi^{\mathsf{r}}_{\mathsf{fresh}}(\cdot)$ re-use existing machinery from fresh
- If f **classical**, there is no problem

Reductionist Rules: Bi-Deduction

Bi-deduction [CSF'22]

Automate simplifications of $\sim$ by **deterministic simulation**.

$$\sharp(u_0, u_1) \triangleright_c \sharp(v_0, v_1) \quad : \quad \exists f : \mathsf{PTM}.\ f(u_0) = v_0 \land f(u_1) = v_1$$

Key rule:

$$\text{deduce}_c \ \frac{u_0 \sim_c u_1 \qquad \sharp(u_0, u_1) \triangleright_c \sharp(v_0, v_1)}{v_0 \sim_c v_1}$$

Example: drop v + compute $((\lambda x.\, H(x))\ u)$

$$\sharp(u_0, u_1), \sharp(v_0, v_1), \lambda x.\, H(x) \ \triangleright_c\ \sharp(u_0, u_1), \mathrm{H}(\sharp(u_0, u_1))$$

$\Rightarrow$ u used twice above, quantum variant unsound

Reductionist Rules: Bi-Deduction

Quantum bi-deduction

Generalization: **deterministic** $\Rightarrow$ **error-free**.

$$\sharp(u_0, u_1) \rhd_{\mathsf{q}} \sharp(v_0, v_1) \quad : \quad \exists f : \mathsf{PQTM_E}.\ \begin{array}{l} f(u_0) = v_0 \wedge \\ f(u_1) = v_1 \end{array}$$

Error-free quantum machines $\mathsf{PQTM_E}$

- Avoid difficulties with **measurement randomness**
  i.e. f independent from (u, v)
- For quantum values, only basic manipulations
  Example: swapping $(c, q) \rhd (q, c)$
- For more complex manipulations:
  automatic deduce + manual fa for **att**$(\cdot)$

Reductionist Rules: Bi-Deduction

Proof system: $\triangleright_q \approx \triangleright_c +$ **linear usage** of quantum values

Example: transitivity

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

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

Adapting the Proof System

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

Reductionist Rules: Cryptographic Bi-Deduction

Cryptographic reduction to game $\mathcal{G} = \sharp(\mathcal{G}_0, \mathcal{G}_1)$

$$v_0 \sim_c v_1 \quad \text{if} \quad \exists S : \mathrm{PPTM}.\ S^{\mathcal{G}_0} = v_0 \ \wedge \ S^{\mathcal{G}_1} = v_1$$

Examples: IND-CCA, PRF, DDH

Classical cryptographic bi-deduction [CCS'24]

$$\cdots \vdash \sharp(u_0, u_1) \triangleright_c^{\mathcal{G}} \sharp(v_0, v_1)$$

- **Complex semantics**: $\mathcal{S}$ probabilistic + $\mathcal{G}$ stateful ($\cdots$)
- **Proof system** for $\triangleright_c^{\mathcal{G}}$
- **Automatic proof-search**, including induction

Reductionist Rules: Cryptographic Bi-Deduction

Challenges

- $\mathcal{S}$ probabilistic $\Rightarrow$ quantum error-free **insufficient**
- Generalize $\mathcal{S}$ 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

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

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

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 (t, out) = {
    r <$ qrand;
    (input,state) = att(r, (state,out));
  }

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

Reductionist Rules: Cryptographic Bi-Deduction

- **Semantics** of $\triangleright_{\mathsf{c}}^{\mathcal{G}}$ unchanged
  (except minor adaptation to have $\mathcal{G}$ quantum)
- **Proof system** unchanged
- **New induction rule** specialized for the quantum execution model

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

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

Case-Studies

Case Studies

Case studies in our post-quantum Squirrel

- Four **KEM combiners (CPA/CCA)**:
  XOR, XOR-then-MAC, Dual-PRF, Nested Dual-PRF
- Two **hybrid key-exchange protocols (strong secrecy)**:
  BCGNP [S&P'22], C_{Sigma}
- Case studies use Squirrel latest features: crypto, smt

Proof strategy

- First proof in classical setting/execution model (≈ pers. months)
- Then, adapted to post-quantum (≈ pers. days)

Conclusion

Conclusion

- **Execution model for PQC**
- **A faithful logic for PQC**
  Adequacy result
- **Adapt Squirrel proof systems**
  Latest features, e.g. smt, crypto
- **Implementation + validation**
  Case-studies: Hybrid KEM Combiners + KE

Thank you for your attention

Execution Model (Simplified)

```
let rec frame (t : timestamp) =
   (state t, transcript t)

and transcript (t : timestamp) =
   ⟨ transcript (pred t), input t, output t ⟩

and state (t : timestamp) =
   att(qrnd (pred t), frame (pred t))#2

and input (t : timestamp) =
   att(qrnd (pred t), frame (pred t))#1

and output (t : timestamp) = ... (* protocol specific, uses input t *)
```