UNIVERSITY OF TARTU 1632
erc
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
EXCITE
Estonian Centre of Excellence in ICT Research
European Union
Regional Development Fund
Investing in your future

Formal Verification of Quantum Cryptography

(Overview of our project)

Dominique Unruh

University of Tartu

Why verify security?

- Crypto protocols – ubiquitous, and potentially high cost of failure

- Bugs “want” to be exploited
  (attacker intentionally tries to exploit)

- Conclusion:
  Security proofs!

The trouble with security proofs

- **In theory:**
  Once a protocol is proven secure,
  it is secure

- **In practice:**
  - Implementation is broken
  - Proof is wrong

Computer-aided verification to the rescue!

Example: Proof in the cryptographic model

$f : D \to D$ one-way permutation

$g(x) := f(f(x))$

Is g one-way permutation?

$$B(y) := f(A(y))$$

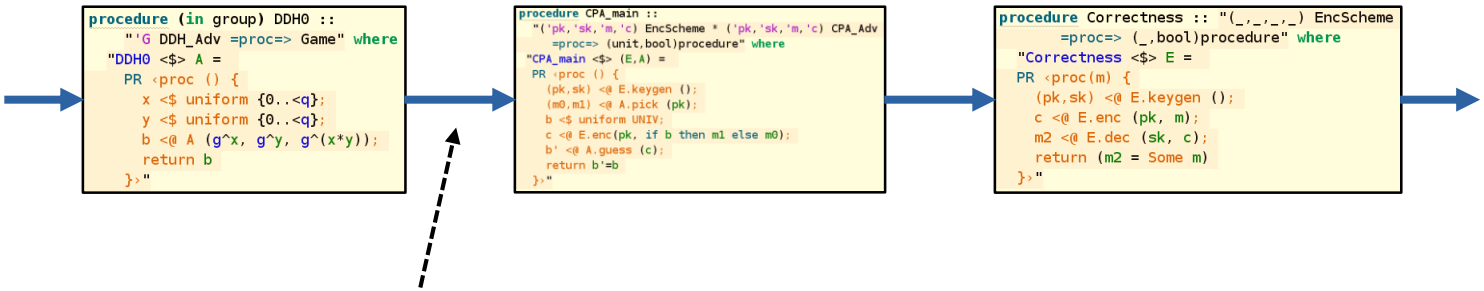
$$\left[\begin{array}{l} z = z' \\ x = x' \\ z \leftarrow D \text{ random} \\ y := f(z) \\ z' := B(y) \end{array} \right] \approx 0$$

g is one-way permutation ✓

Crypto proofs – more formally

“Sequences of Games”

(established approach for crypto proofs)



How to prove relationship?

How to find sequence of games?

Important insight

Crypto verification boils down to reasoning about programs

(E.g., Hoare logics and similar)

Relational Hoare Logic (RHL)

- Describes relation of two programs
- How do the variables of the two programs relate?

$$\{x = y\} \quad x \coloneqq x + 1 \sim z \coloneqq y \quad \{x = z + 1\}$$

- Used, e.g., in EasyCrypt for classical verification (using a probabilistic variant)

Our vision & research

Computer-aided verification

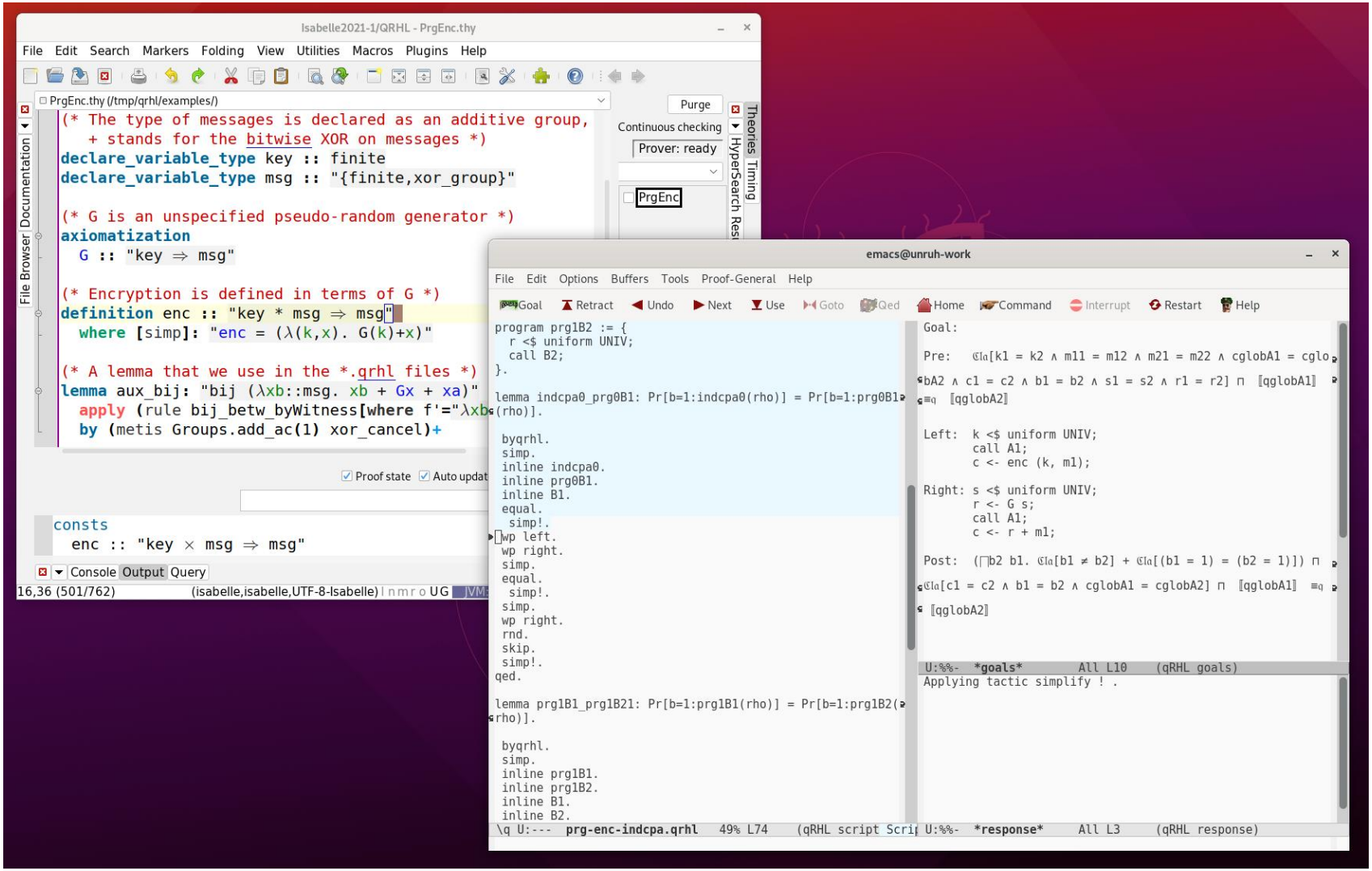
- of post-quantum crypto
- of quantum protocols

Design of interactive tools,
automation where possible

Why not the same logics?

- Many proofs similar in class/quantum setting
- But: Some classical proofs use quantum-unsound techniques:
  - Copying state
  - Fixing coins
- Generally, EasyCrypt is not sound for quantum!

qrhl-tool


Isabelle2021-1/QRHL - PrgEnc.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
PrgEnc.thy (/tmp/qrhl/examples/)
(* The type of messages is declared as an additive group,
+ stands for the bitwise XOR on messages *)
declare_variable_type key :: finite
declare_variable_type msg :: "{finite,xor_group}"
(* G is an unspecified pseudo-random generator *)
axiomatization
G :: "key ⇒ msg"
(* Encryption is defined in terms of G *)
definition enc :: "key * msg ⇒ msg"
where [simp]: "enc = (λ(k,x). G(k)+x)"
(* A lemma that we use in the *.qrhl files *)
lemma aux_bij: "bij (λxb::msg. xb + Gx + xa)"
apply (rule bij_betw_byWitness[where f'="λxb
by (metis Groups.add_ac(1) xor_cancel)+
Purge
Continuous checking
Prover: ready
PrgEnc
Theories
Timing
HyperSearch Res
File Browser
Documentation
Proof state
Auto updat
consts
enc :: "key × msg ⇒ msg"
Console Output Query
16,36 (501/762)
(isabelle,isabelle,UTF-8-Isabelle)
emacs@unruh-work
File Edit Options Buffers Tools Proof-General Help
Goal Retract Undo Next Use Goto Qed Home Command Interrupt Restart Help
program prg1B2 := {
r <$ uniform UNIV;
call B2;
}.
lemma indcpa0_prg0B1: Pr[b=1:indcpa0(rho)] = Pr[b=1:prg0B1
(rho)].
byqrhl.
simp.
inline indcpa0.
inline prg0B1.
inline B1.
equal.
simp!.
wp left.
wp right.
simp.
equal.
simp!.
simp.
wp right.
rnd.
skip.
simp!.
qed.
lemma prg1B1_prg1B21: Pr[b=1:prg1B1(rho)] = Pr[b=1:prg1B2(
rho)].
byqrhl.
simp.
inline prg1B1.
inline prg1B2.
inline B1.
inline B2.
\q U:--- prg-enc-indcpa.qrhl 49% L74 (qRHL script Scri
Goal:
Pre: ℭ𝔩𝔞[k1 = k2 ∧ m11 = m12 ∧ m21 = m22 ∧ cglobA1 = cglo
bA2 ∧ c1 = c2 ∧ b1 = b2 ∧ s1 = s2 ∧ r1 = r2] ⊓ ⟦qglobA1⟧
≡q ⟦qglobA2⟧
Left: k <$ uniform UNIV;
call A1;
c <- enc (k, m1);
Right: s <$ uniform UNIV;
r <- G s;
call A1;
c <- r + m1;
Post: (⨅b2 b1. ℭ𝔩𝔞[b1 ≠ b2] + ℭ𝔩𝔞[(b1 = 1) = (b2 = 1)]) ⊓
ℭ𝔩𝔞[c1 = c2 ∧ b1 = b2 ∧ cglobA1 = cglobA2] ⊓ ⟦qglobA1⟧ ≡q
⟦qglobA2⟧
U:%%- *goals* All L10 (qRHL goals)
Applying tactic simplify ! .
U:%%- *response* All L3 (qRHL response)

Quantum Relational Hoare Logic (qRHL)

- Quantum variables X, Y

Elementary while-language

Subspaces

$$\{X = Y\} \quad apply\ U\ to\ X$$
$$\sim\ apply\ U\ to\ Y \quad \{X = Y\}$$

- Where are the subspaces in the example???

Definition of qRHL

$$\{X = Y\} \quad apply\ U\ to\ X \sim apply\ U\ to\ Y \quad \{X = Y\}$$

separable

For any quantum state that satisfies this

After applying these programs

separable

There exists a new state with same marginals satisfying this

- Without "separable", cannot prove "frame rule" (for modular reasoning)

Quantum Equality

Equal rule: $\{X_1 \equiv X_2\}\, \boldsymbol{c} \sim \boldsymbol{c}\, \{X_1 \equiv X_2\}$

- What does "$X \equiv Y$" mean?
- Quantum variables have no individual content (due to entanglement)
- Need definition that is subspace

$|\Psi\rangle$ satisfies $X_1 \equiv X_2$ iff

$|\Psi\rangle$ invariant under swapping $X_1 \leftrightarrow X_2$

Case study: Fujisaki-Okamoto

- Assuming passively secure encryption
- Make actively secure encryption (KEM)

$\underline{\mathsf{Encaps}(pk)}$

01 $m \leftarrow_{\$} \mathcal{M}$
02 $c := \mathsf{Enc}(pk, m; \mathsf{G}(m))$
03 $K := \mathsf{H}(m)$
04 **return** (K, c)

$\underline{\mathsf{Decaps}(sk, c)}$

05 $m' := \mathsf{Dec}(sk, c)$
06 **if** $m' = \perp$ **or** $\mathsf{Enc}(pk, m'; \mathsf{G}(m')) \neq c$
07 **return** $K := \mathsf{H}_{\mathsf{r}}(c)$
08 **else return** $K := \mathsf{H}(m')$

- Many variants… This one proven by Hövelmanns, Kiltz, Schäge, Unruh

Challenges

- Verification conditions:
  - Products of operators on different subsystems
  - E.g., $CNOT_{XY} \cdot CNOT_{YZ}$
  - Very cumbersome to reason about formally
- Quantum equality:
  - $X_1 Y_1 \equiv X_2 Y_2$ is not $X_1 \equiv X_2 \wedge Y_1 \equiv Y_2$
- No automation for common tasks
- Lots of copy&paste

Solutions

- Cumbersome reasoning about VCs:
  - "Register formalism"
  - Treats quantum registers as mathematical objects

```
‹⟦q2, r2⟧ =q EPR ⊓ ⟦q1, r1⟧ =q EPR
 ≤ hadamard»⟦r2⟧ *s hadamard»⟦q1⟧ *s ⟦q1, r1⟧ ≡q ⟦q2, r2⟧›
```

  - I believe this will make things work considerably better (in progress)
- Copy & paste
  - Module system for programs (not started)

Register formalism

- What is a quantum register?
- "Pointer" to a location in quantum memory
  - E.g., index, name
- Too narrow for convenience
  (If x, y are registers, xy is not.)
- Solution: Abstract notion of a register
  - xy is register
  - $x.z$ is a register (z inside x)

How are registers defined?

- Register X from $\mathcal{H}_{reg} \to \mathcal{H}_{mem}$
- Described by a unital *-homomorphism $L(\mathcal{H}_{reg}) \to L(\mathcal{H}_{mem})$
  - (Extra subtleties for infinite-dimensional case)
- Given an operation U on $\mathcal{H}_{reg}$, $X(U)$ is corresponding op. on $\mathcal{H}_{mem}$
- Everything else can be constructed from this

Isabelle/HOL backend

- Allows free reasoning about VCs
  (not constrained by the qRHL rules in the tool)
- Long-term goal:
  - Have everything fully formalized in Isabelle/HOL
    (no trusted axioms)

Isabelle/HOL formalization

- Complex bounded operator formalization
- Registers
  - Finite-dimensional
  - Infinite: WIP
  - Integration in qrhl-tool: WIP
- Tensor products
  - Finite-dimensional
  - Infinite: WIP
- qRHL semantics and logic: missing

Crypto proofs in qrhl-tool

- Small examples
- Fujisaki-Okamoto
- WIP: Compressed quantum oracles

- Want:
  - NIST candidates
  - Actual quantum protocols (QKD?)

Further missing things

- Automation / decision procedures?
- Module system?
- Connection to EasyCrypt?

Postdoc/phd at University of Tartu:

Verification of Quantum Cryptography

European Research Council
Established by the European Commission

http://tinyurl.com/postdoc-vqc

- Quantum logic?
- Thm proving?
- Q info-theo/crypto?

TARTU ÜLIKOOL · UNIVERSITAS TARTUENSIS · 1632