Individual Position Slides: Jonathan Katz

(University of Maryland)

(Apologies I can’t be here in person)

Impedance mismatch

- I’d like to echo what Jon Herzog said (though in slightly different terms)
  - Most of the protocol analysis in EasyCrypt had *nothing* to do with “cryptography”; instead, it involved manipulating data structures
- But…it’s hard to “blame” EasyCrypt
  - The protocols in question were “cryptographically simple” but “data structure”-heavy
  - In particular:
    - Relatively small fraction of the proofs relied on computational assumptions
    - Definitions themselves were complex
  - Not clear that these are the types of protocols EasyCrypt should be targeting
    - What are the crypto primitives/proofs that EC should be targeting?

Impedance mismatch

- Nevertheless, would be nice if EasyCrypt offered better support for “modularity”
  - This is how cryptographers build complex protocols
  - This is how cryptographers reason about complex protocols
  - This is how cryptographers prove security of complex protocols
- Not clear (to me) whether instantiation will fully address this

Other comments

- A “wish list” for EasyCrypt

- Some musings on formal verification in general

- Theme: A formal proof is only as good as…
  - …your hidden assumptions
  - …your definitions/cryptographic assumptions
  - …your axioms
  - …how faithfully your EC code captures your implementation

Running time

- EasyCrypt has no way to reason about running time
  - Nothing prevents a reduction from computing discrete logarithms
  - Nothing prevents a (human) proof verifier from believing such a proof

- Is this an issue?
  - Practically speaking?
    - Not in general (but there is always the chance of unintentional error)
    - For some proofs, however, analysis of the running time of the reduction is non-trivial (e.g., zero-knowledge simulators)
  - Formally speaking? Yes

- Unclear how to encode the notion of “polynomial time” in EasyCrypt, which does not deal with asymptotics at all

Definitions/assumptions

- In the course of doing a reductionist security proof, it can become difficult (non-obvious) to verify that you are proving the right thing/reducing to the right assumptions

- Would be *extremely* useful to have a library of “standard assumptions” included as part of the EasyCrypt distribution, that could be accessed as “black boxed”
  - Proofs would reduce to *the* Diffie-Hellman assumption, rather than my (possibly buggy) version of the Diffie-Hellman assumption
  - I would prove CPA-security, rather than my (possibly buggy) version of CPA-security

Axioms

- Incorrect/inconsistent axioms can allow you to prove anything

- Unclear what to do about this in general
  - Verifying all axioms in Coq does not seem viable

- Two partial suggestions
  - Periodically check whether possible to prove 0=1
    - Alert user in that case
  - Include “standard axioms” on strings, groups, etc. as part of EasyCrypt distribution
    - Manual review; could be verified in Coq over time

Protocol vs. implementation

- Would be nice to know that the protocol you are proving secure matches the protocol you are implementing

- Future research directions:*
  - Compiler from, e.g., (subset of) C code to EasyCrypt code
  - Provide better “syntactic sugar” in EasyCrypt

- Would also reduce the burden on the user

* This may already be done; I am not sure

Protocol vs. implementation

- In fact, even if one is careful there can be a mismatch between the protocol you are proving secure and the protocol implementation

- Example:
  - In EasyCrypt, group elements might have type `group`
  - In your implementation, group strings might be byte arrays
  - These are not the same thing!
    - E.g., anything of type `group` is guaranteed to be a group element, but not every byte array is necessarily a valid encoding of a group element; cf. small-subgroup attacks
    - Other examples, too

Parting thoughts

- Crypto protocols/proofs becoming ever more complex
  - Unfortunately, many proofs never written at “journal-quality” level
  - (Many proofs never written at any reasonable level)
  - Unfortunately, most proofs never verified before publication
  - (Many proofs never verified at all)

- “Would be nice if all published crypto papers came with machine-verified proofs of security”
  - We are not even close to making this viable (yet)

- What are the proofs that EasyCrypt should be targeting?