Panel: perspectives and experiences applying EasyCrypt 'outside' the laboratory

First EasyCrypt Workshop

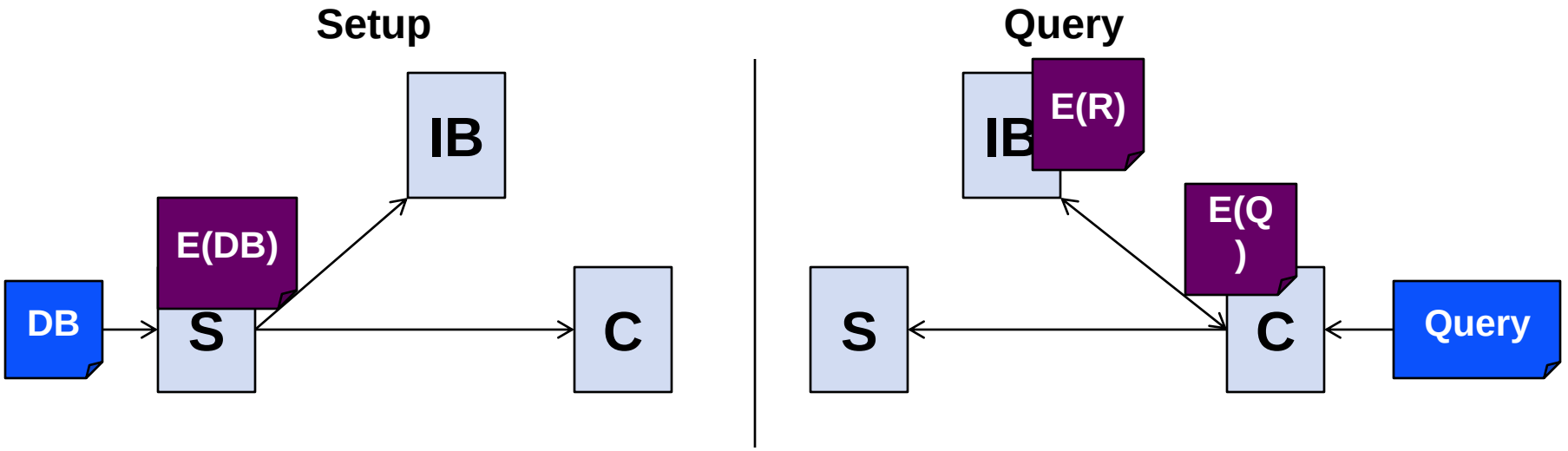
19 July 2013

This work is sponsored by the Director of National Intelligence under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Overview

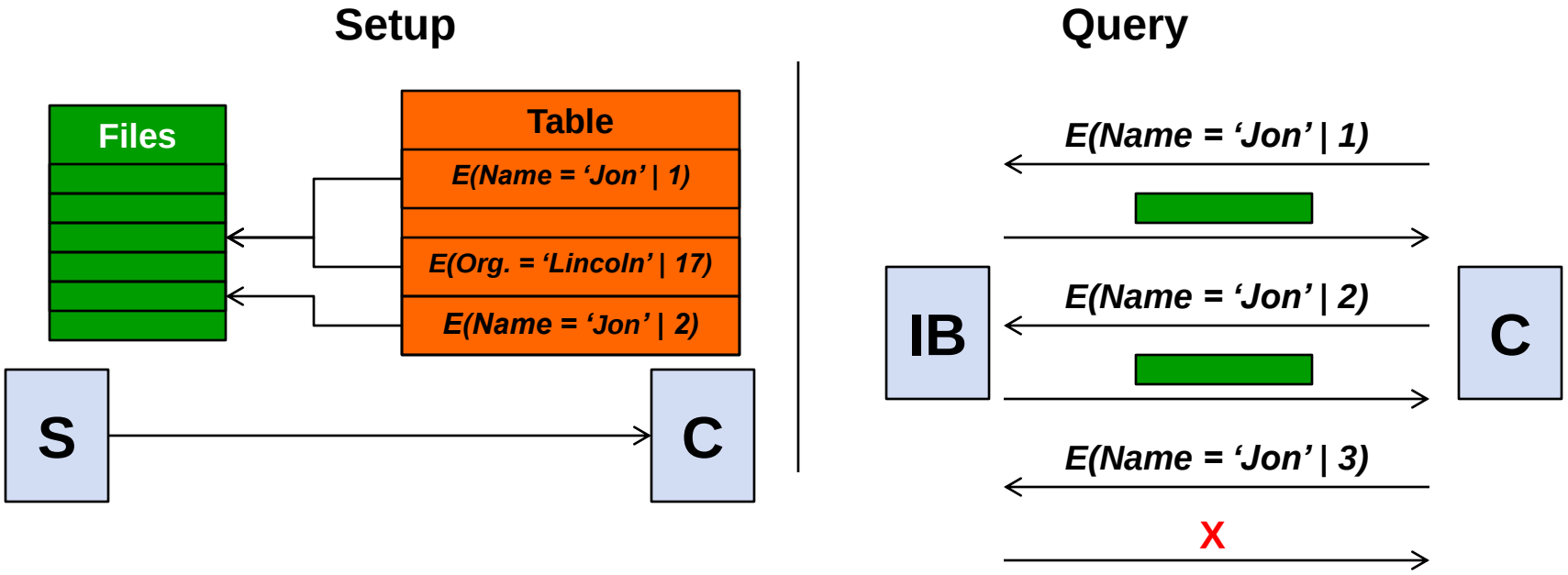
- **Basis for this panel: applying EC to a large protocol**
  - **Large team of cryptographers, formal-methods experts**
    - **No EC developers**
  - **Effort duration of 12 – 18 months**
  - **Attempt to verify 3 security properties from 8-page proof**
- **Successes, new techniques already described in prior (summer school) lecture**
- **This panel: challenges and suggested future directions**
  - **‘Impedance mismatch’ between EC and crypto research**
  - **Understanding provability and truth in EasyCrypt**
  - **The need to tame complexity**
  - **The limits of verification**
- **First, some background**

Privacy-Preserving Information Retrieval



- **Goal: Allow client to query server's database, privately**
  - **Server should not learn query**
  - **Client should not learn database (other than answer to query)**
- **Usually implemented with 'isolated box' (third party)**
  - **Setup: Server sends encrypted DB to IB (plus other communication)**
  - **Client sends encrypted query to IB (plus other communication)**
  - **IB responds with encrypted matching records**

The ‘UC Irvine’ protocol*



- Two *encrypted* data structures
  - Files: encrypted records
  - Table: masked pointers into Files for each valid query
    - To avoid correlations between entries, Table rows given unique ‘counters’
- Protocol: Client makes successive masks for given query
  - Continues sending masks, getting records, until failure

* Privacy-Protecting Information Retrieval, University of Irvine Team: Protocol and Proofs.
Included in Appendix E of Security and Privacy Assurance Research (SPAR) Program Broad Agency Announcement
https://www.fbo.gov/utils/view?id=32750071e5cf4afc3b7e973d608e657e

Notable characteristics and security properties

- **Security properties: the view of each participant should be simulatable given only limited information**
  - **Server's view simulatable from database and the number of queries**
  - **Client's, IB's views similarly simulatable**
- **All proofs: 'real' game indistinguishable from simulated game**
- **Technical ingredients:**
  - **Random oracles, hash functions, and symmetric encryption**
  - **Compound data-structures: databases, queries**
  - **'For' loops over lists (database, Files, Table, etc.)**
  - **'While' loops (client-IB interaction during query)**
  - **Random permutations on lists**
- **Protocols and proofs complex, but not unusually so**

Analysis efforts

- **UC Irvine protocol developed as part of U.S. Gov't project**
  - **Manually reviewed by academic cryptographers**
- **Follow-on project (SPAR) also included new crypto & review**
- **Also included 6-person formal-methods team**
  - **MIT Lincoln Laboratory and Naval Research Laboratory**
  - **Variety of backgrounds: computation crypto, formal methods/PL, CryptoVerif, Dolev-Yao model…**
- **Attempted to verify UC Irvine protocol in EC**

Results (1/2)

- ‘Mini UC Irvine’ (simplified version): 100% complete
- Actual server: 13 games, 4K lines of code, 80% finished
  - 1 prior sequence of games completed, scrapped
- Actual IB: 10+ games, 7.5K lines of code, 66% (?) finished
  - 1 prior sequence of games scrapped after 4 months
  - Unclear how to finish current sequence of games
    - Upcoming transition depends on high-level fact about sets (P-I-E)
- Client: 10 games, ~6K lines of code, 80% finished
  - 500 LOC per completed transition
  - 2 prior proof-sequences scrapped after 4-6 months
  - Will probably need to scrap current game-sequence, too
    - Last transition sound, but no clear way to prove in EasyCrypt

Results (2/2)

- **Main successes (new techniques, 'mini UCI' analysis) already presented in summer-school lecture**
- **Other successes:**
  - **EC helped us find bugs that crept past manual review**
    - **Simulator shuffles DB, real server does not**
    - **Proof written in concrete model, but uses deterministic hash functions**

    Both flaws easily patched, but found due to EC's formalism
  - **Feedback to EasyCrypt developers (modules, instantiation)**
  - **Larger protocols, proofs than previously attempted in EasyCrypt**

- **This panel: constructive look at the negatives**
  - **What were the technical obstacles to our analysis?**
  - **How can these be mitigated?**
  - **Where should EC development go?**

This panel: perspectives and experiences

Alley Stoughton
(MIT Lincoln Laboratory)

Cathy Meadows
(Naval Research Laboratory)

Aaron Jaggard
(Naval Research Laboratory)

Jon Katz
(University of Maryland)

Jon Herzog
(MIT Lincoln Laboratory)

Adam Petcher
(MIT Lincoln Laboratory)

Note: panelists will present their own opinions

Individual position slides: Jonathan Herzog

Obstacles to success

- **Disclaimer: analysis hindered due to late-in-game ‘course correction’**
  - **Hashing changed from deterministic function to random oracles**
  - **Some prior progress lost, some game-sequences scrapped**
- **But other problems more general, more relevant, and ultimately more important***
- **Chose two to discuss here:**
  - **Specific technical obstacles**
  - **‘Impedance mismatch’ between EC and actual practice of cryptographic proofs**

*** My opinion**

Technical problems

- No way to invoke the generic definition of lower-level primitive
  - 'This transition is a special case of semantic security'
  - Soon moot: 'instantiation' to be provided in future version of tool
- Tracking probability-distributions: how to show these to be equivalent?

```
for (i = 0 to n) {
        key[i] = generate_key();
        ct[i] = enc(key[i],
msg[i]);
        }
```

```
for (i = 0 to n) {
        key[i] = generate_key();
        }
for (i = 0 to n) {
        ct[i] = enc(key[i],
msg[i]);
        }
```

- The 'permutation problem: how to show these are the same?

```
for (i = 0 to n){
        key[i] = generate_key();
        }
```

```
perm = random_permutation(n)
for (i = 0 to n){
        j = perm(i);
        key[j] = generate_key();
        }
```

‘Impedance mismatch’

- **(My) primary problem actually not technical:**
  - **Huge disconnect between EC ‘mindset’ and my actual background**
- **Computational cryptography an incomplete preparation**
  - **Good for devising reasonable-seeming sequence of informal, human-readable games**
  - ***Not at all* good for proving sequence of games in EC**

Typical problem

```
Current goal
Pending subgoals: 1
pre   = true
stmt1 =   1 : (i, l) = A ();
stmt2 =   1 : (i, l) = A ();
          2 : perm = randomPermutation (length (l));
          3 : l = permute (l,perm);
post  = mem (i{1},l{1}) = mem (i{2},l{2})
```

- `auto` **doesn't work.**
- **…Now what?**
  - **Is goal true? Is there a way to prove it from existing lemmas? Do I need to let the SMT solvers run longer? Do I need to add more lemmas? Do I need to add an axiom? How will I be sure it is consistent? Etc. etc. etc.**

Big picture / takeaways

- ***Not*** **saying that EC should require no thought/expertise at all**
- ***Am*** **saying that <u>none</u> of this was covered in my PhD program**
  - **In fact, was implicitly taught to elide over this type of detail**

- ***Predict other working cryptographers will be baffled at how hard it is to prove 'obvious' (i.e., non-cryptographic) steps***
- **Problem at both tactical and strategic levels:**
  - **Unable to answer "how do I prove this specific goal?"**
  - **Also unable to answer "Is this a reasonable sequence of games?"**

- **Need better tactics or library of examples**
  - **To extent possible, allow cryptographers to use EasyCrypt without a second degree in program verification**