Information Security – Theory vs. Reality

0368-4474, Winter 2015-2016

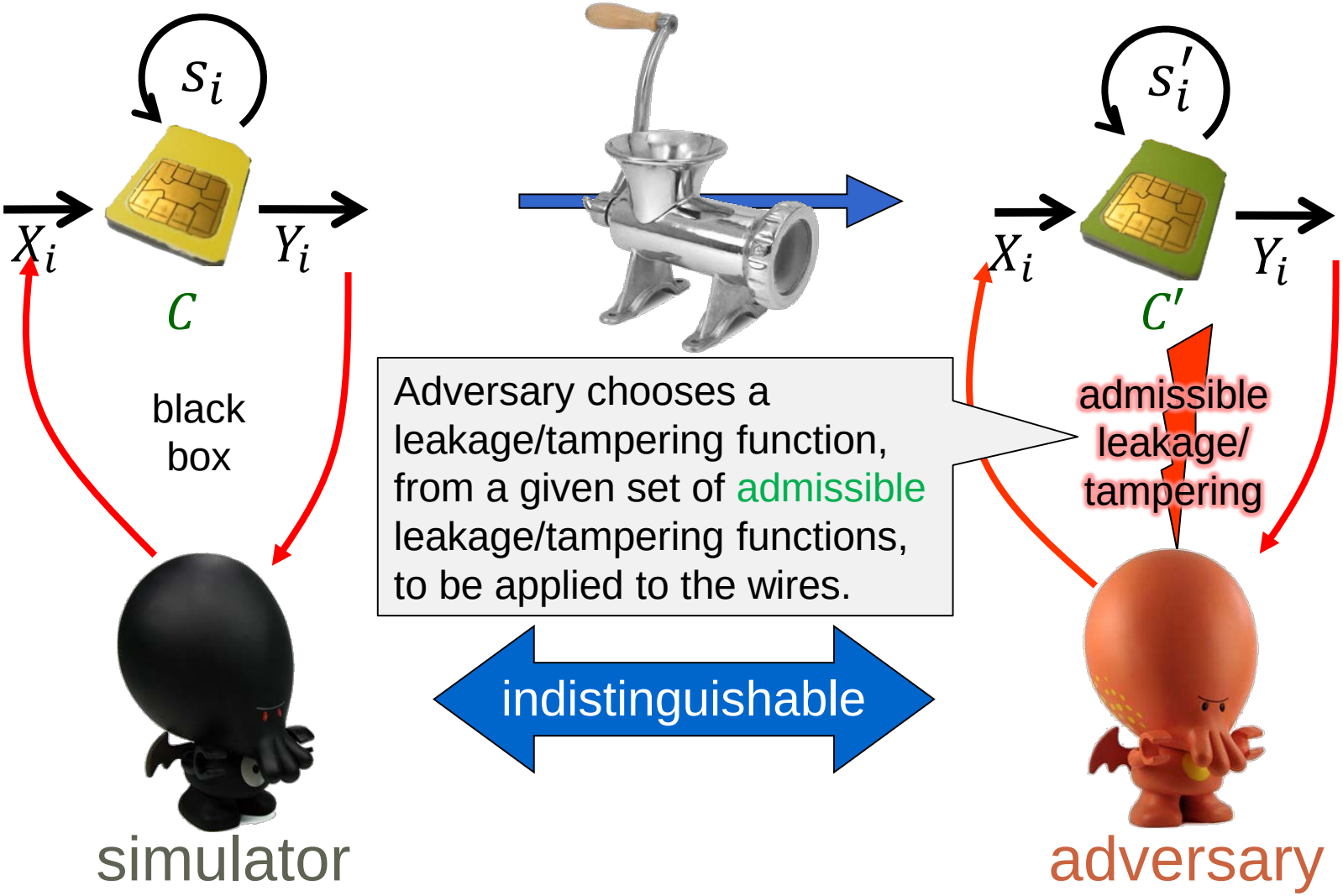
Lecture 9: Leakage resilience (continued)

Lecturer:
Eran Tromer

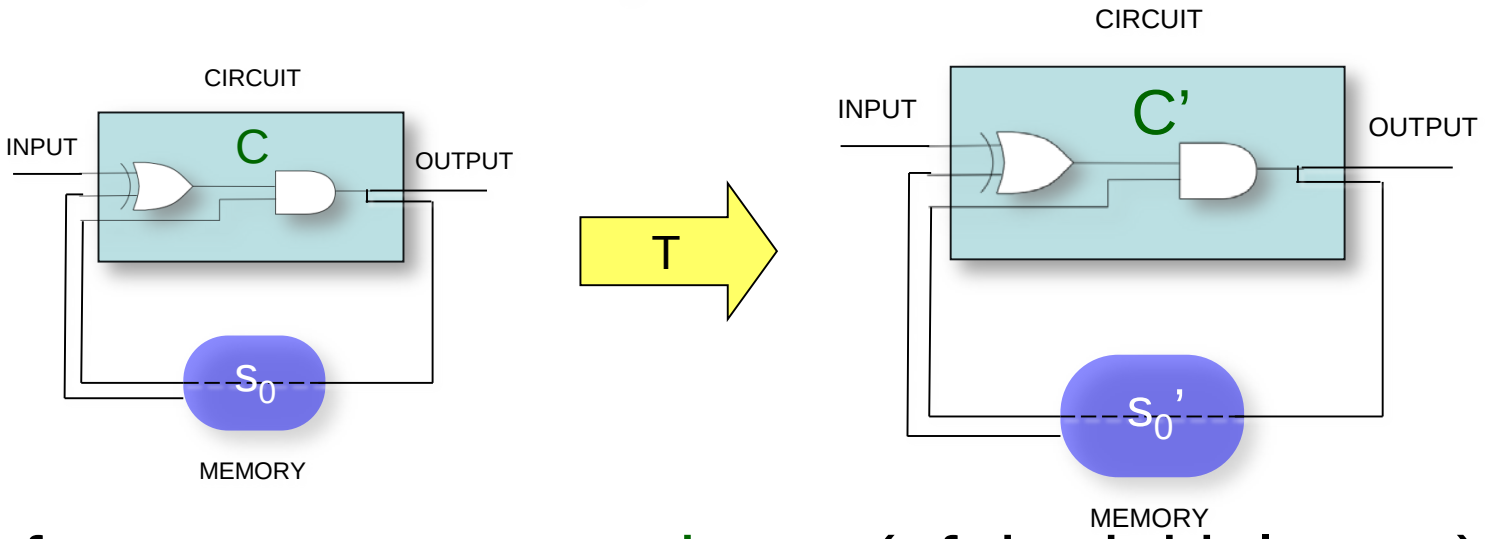
Leakage resilience
(continued)

Security

[Ishai Sahai Wagner '03]



Security definition



Transformer T protects privacy (of the initial state) against a given class of admissible leakage/tampering:

∀circuit C

∃efficient Sim

∀admissible Adv

∀initial state s_0 :

$Sim^{Adv,C[s0]} \approx$ output of Adv attacking C'[s_0']

(Even in case of tampering, only privacy is required)

Resilient-schemes 1/3
(whiteboard discussion)

- Sum-of-wires leakage
  - Dual-Rail Logic
- Sum-of-wire-transitions leakage
  - Dual-Rail Precharge Logic

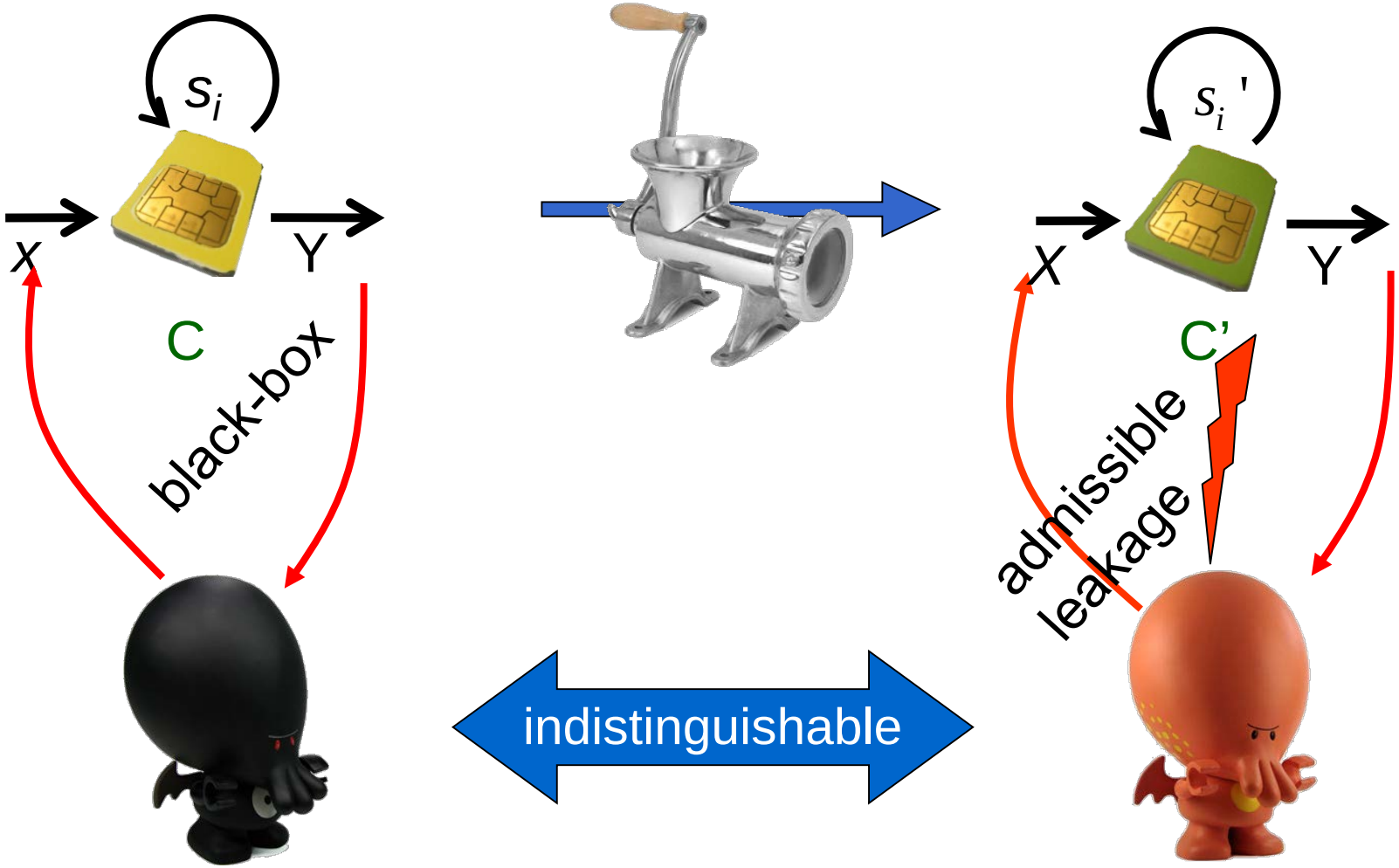
Resilient-schemes 2/3
(whiteboard discussion)

- Single-wire leakage
  - Bit masking or secret sharing
- Multiple-wire leakage
  - Secret sharing
- Leakage of “data-dependent” values from “bulk” computation
  - RSA blinding

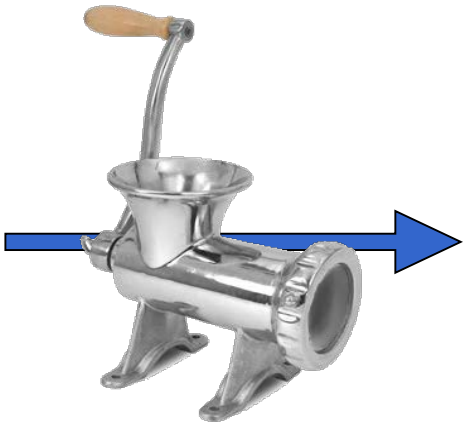
t-wire leakage [ISW03]

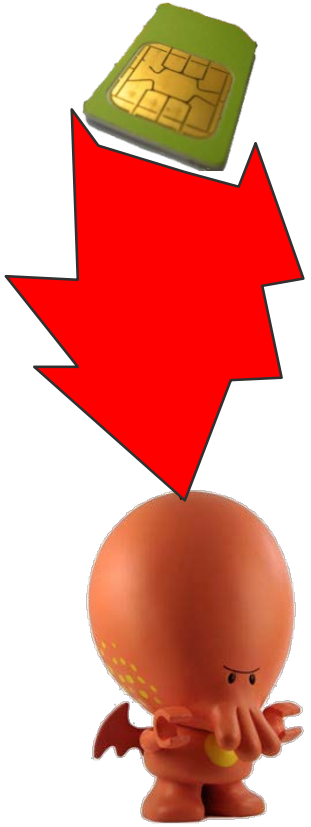
- Secrets additively secret-shared into $m=2t+1$ shares
- Given shares of
  $a=a_1 \oplus \ldots \oplus a_m$ and
  $b=b_1 \oplus \ldots \oplus b_m$:
  - Compute shares of NOT(a) : apply NOT to a_1
  - Compute shares c_i of a AND b :
    - Let $z_{i,j}$, $i<j$, be random independent bits
    - Let $z_{j,i}=(z_{i,j} \oplus a_i b_j) \oplus a_j b_i \quad (i<j)$
    - Let $c_i=a_i b_i \oplus \bigoplus_{j \neq i} z_{i,j}$
- Re-randomize s' at every iteration (hence $m=2t+1$).
- Security proof sketch: simulator runs adversary and, when asked for leakage value: if answer implied by inputs / inputs / previous answer, answers thus. Otherwise answers randomly. This has the correct distribution.

Other leakage?



Our goal





Allow stronger leakage.

Leakage classes

- Locality assumptions
  - Single wire, *t* wires
  - Separate sub-circuits
  - Leak-free processor:
    Oblivious RAM [Goldreich Ostrovsky 95]
  - Leak-free memory ("only computation leaks information" [Micali Reyzin 04]:
    leakage is only from CPU state and memory accessed at that program step)
- Quantitatively bounded
  - Total #bits leaked
  - Total #bits leaked per "computational step"
  - Noisy leakage from every wire
- "Simple leakage"
  - Sums and Hamming weights
  - Low-complexity global functions
- "Too-complicated leakage" (hard to invert)
- *Some of these are for specific functionality (mainly crypto)*

Open problem: realistic models allowing secure and efficient constructions.

Trusted Computing Architecture
(warmup discussion, see next week's slides)