

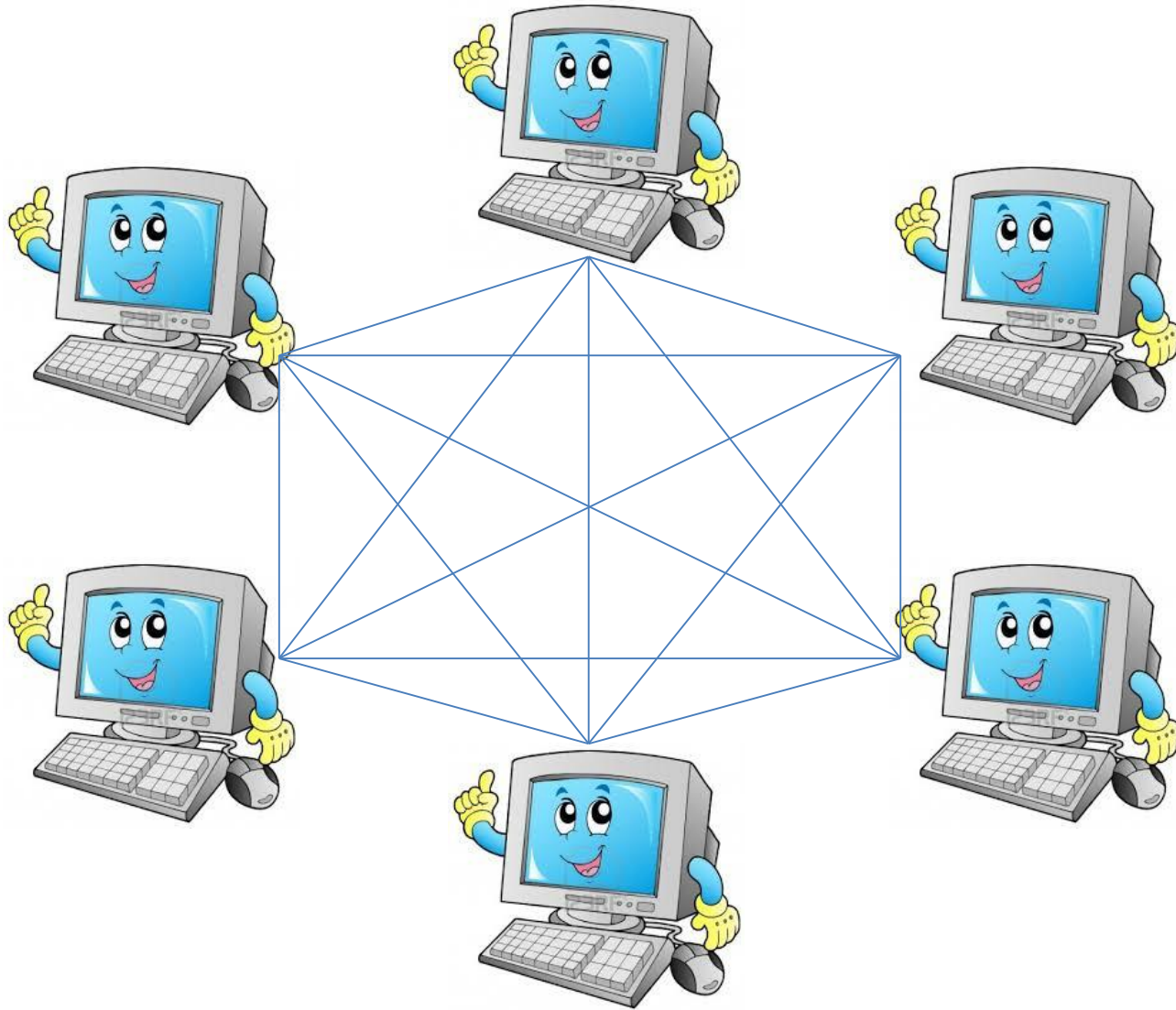
On Adaptively Secure Multiparty Computation with a Short CRS

[SCN'16]

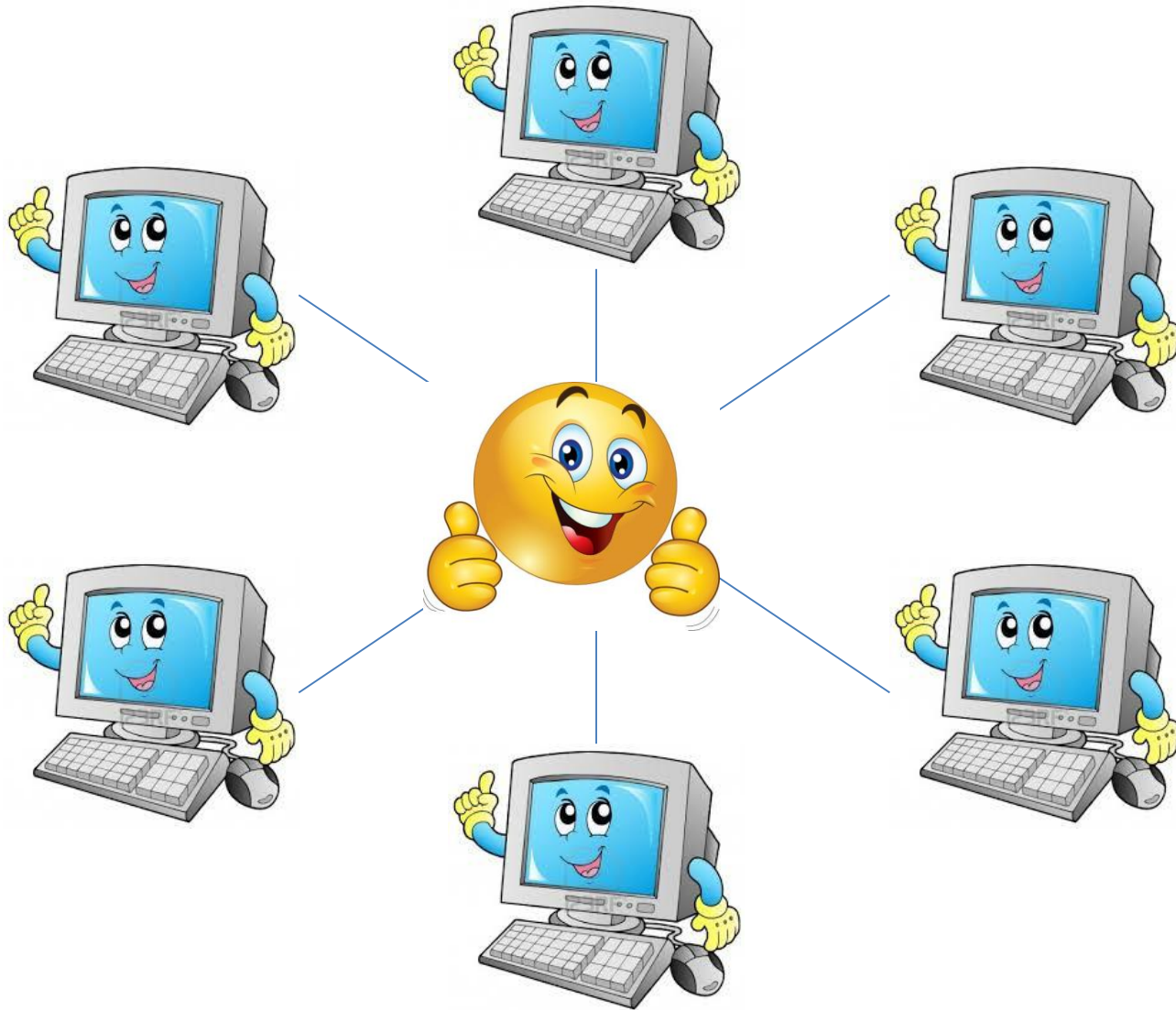
Ran Cohen (Tel Aviv University)

Chris Peikert (University of Michigan)

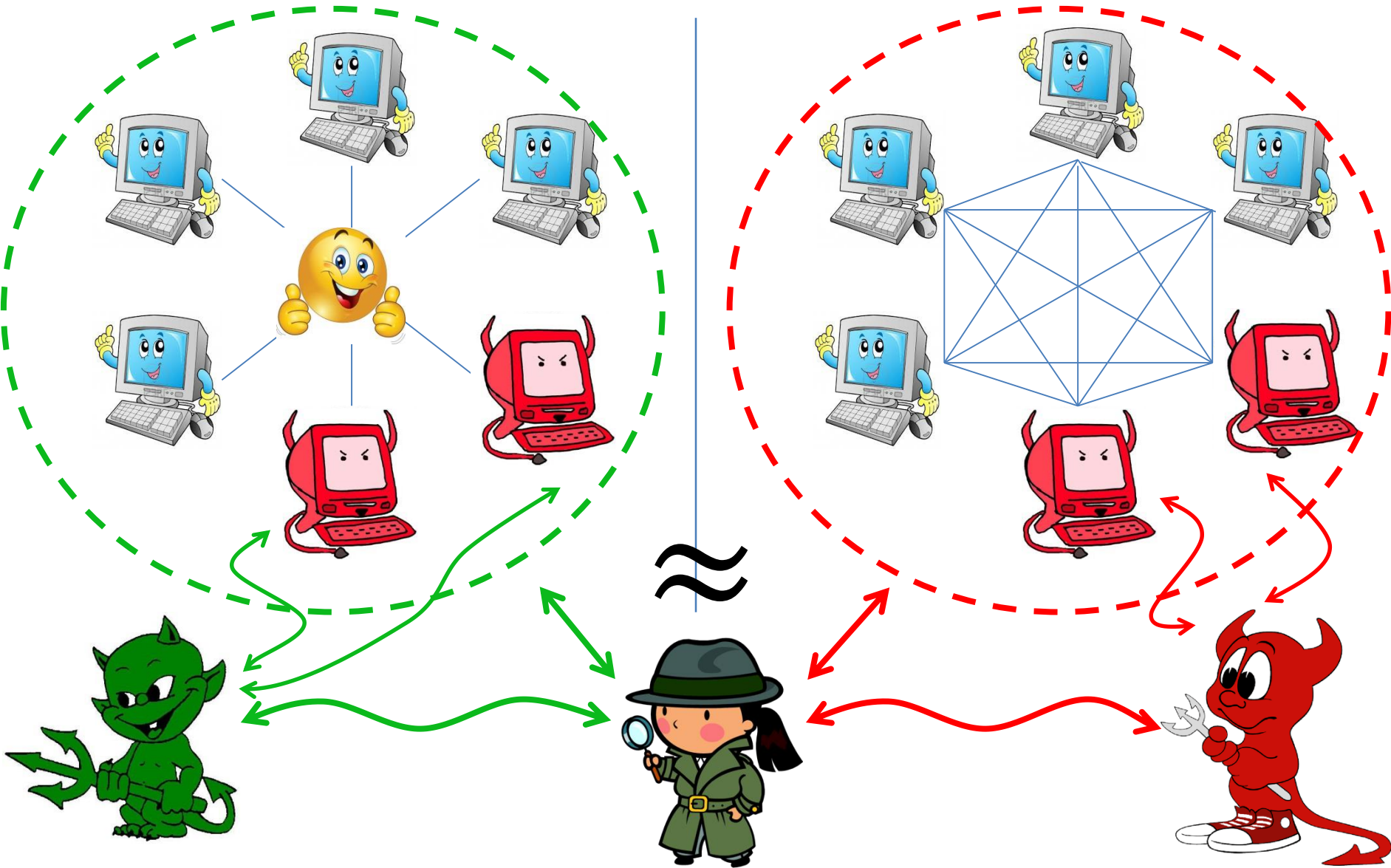
Secure Multiparty Computation (MPC)



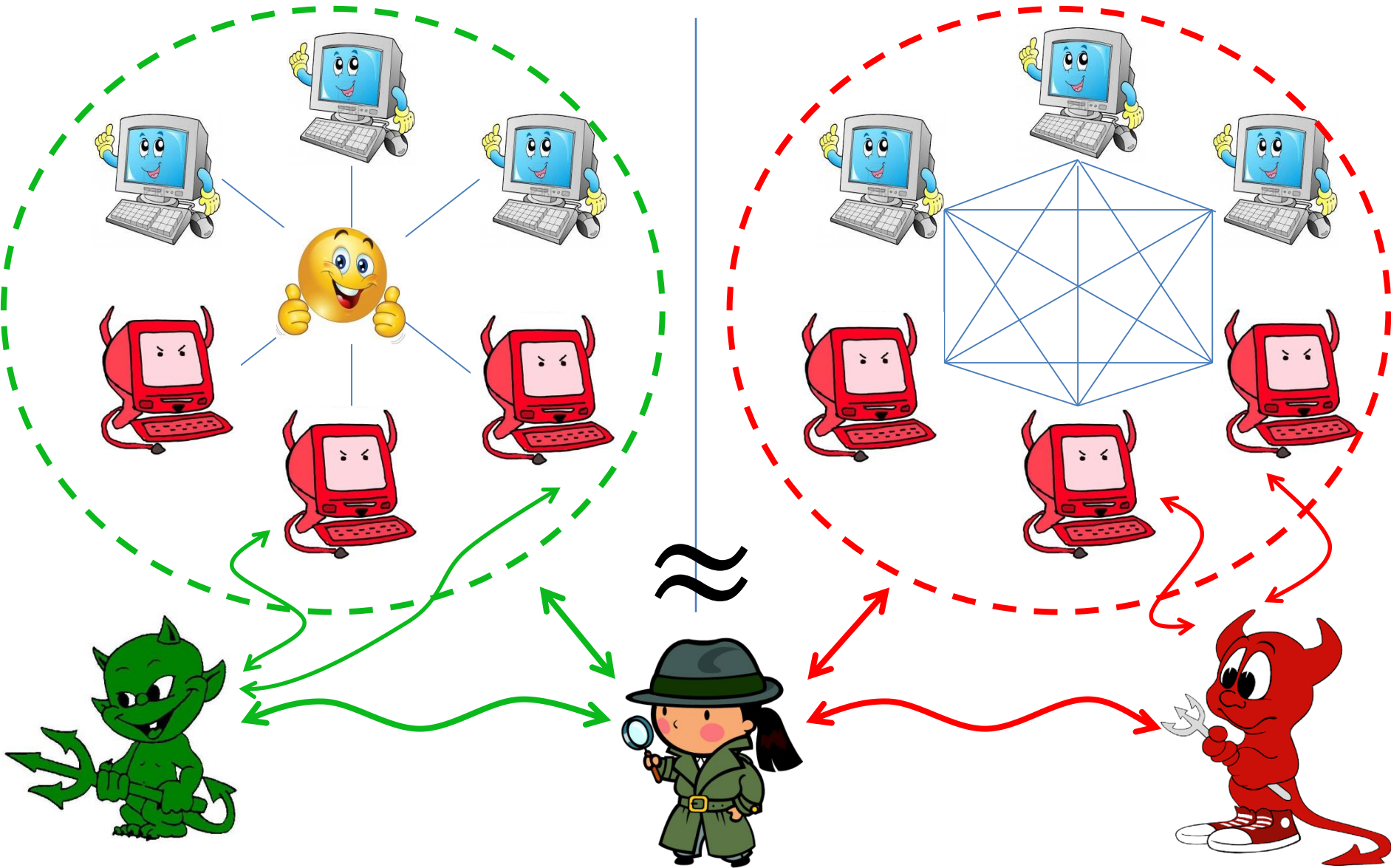
Ideal World/“Functionality”



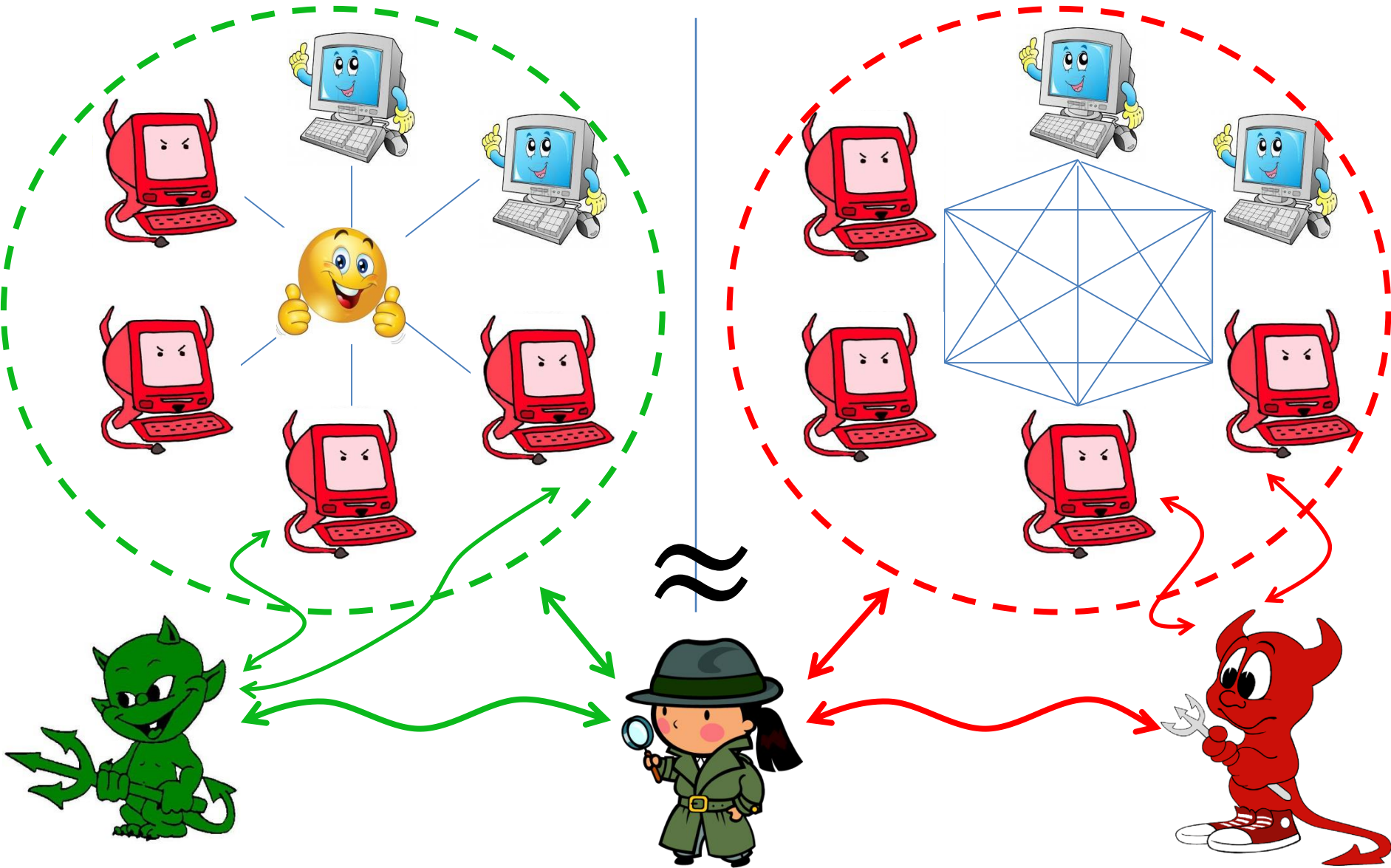
Simulation-Based Security



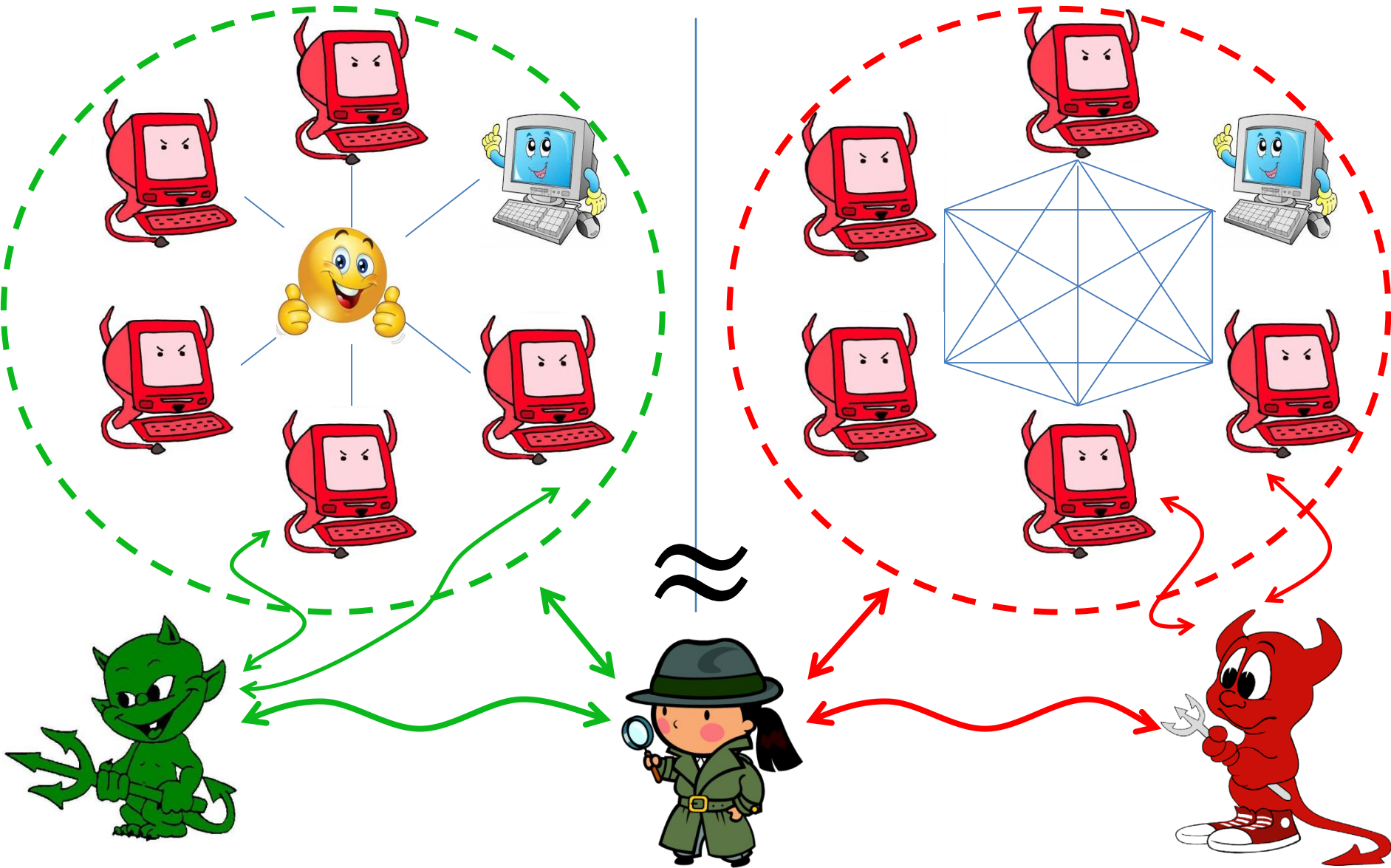
Simulation-Based Security



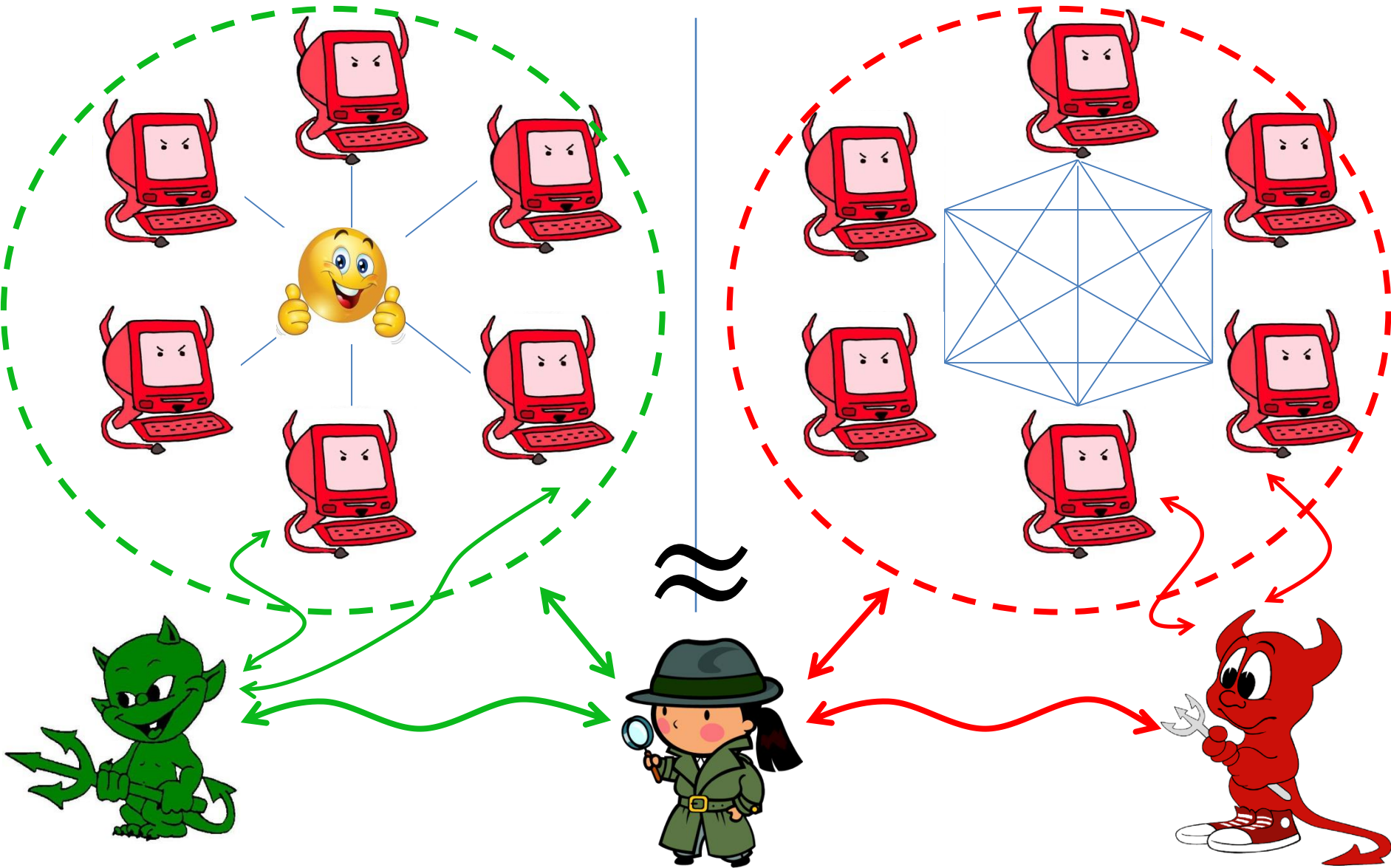
Simulation-Based Security



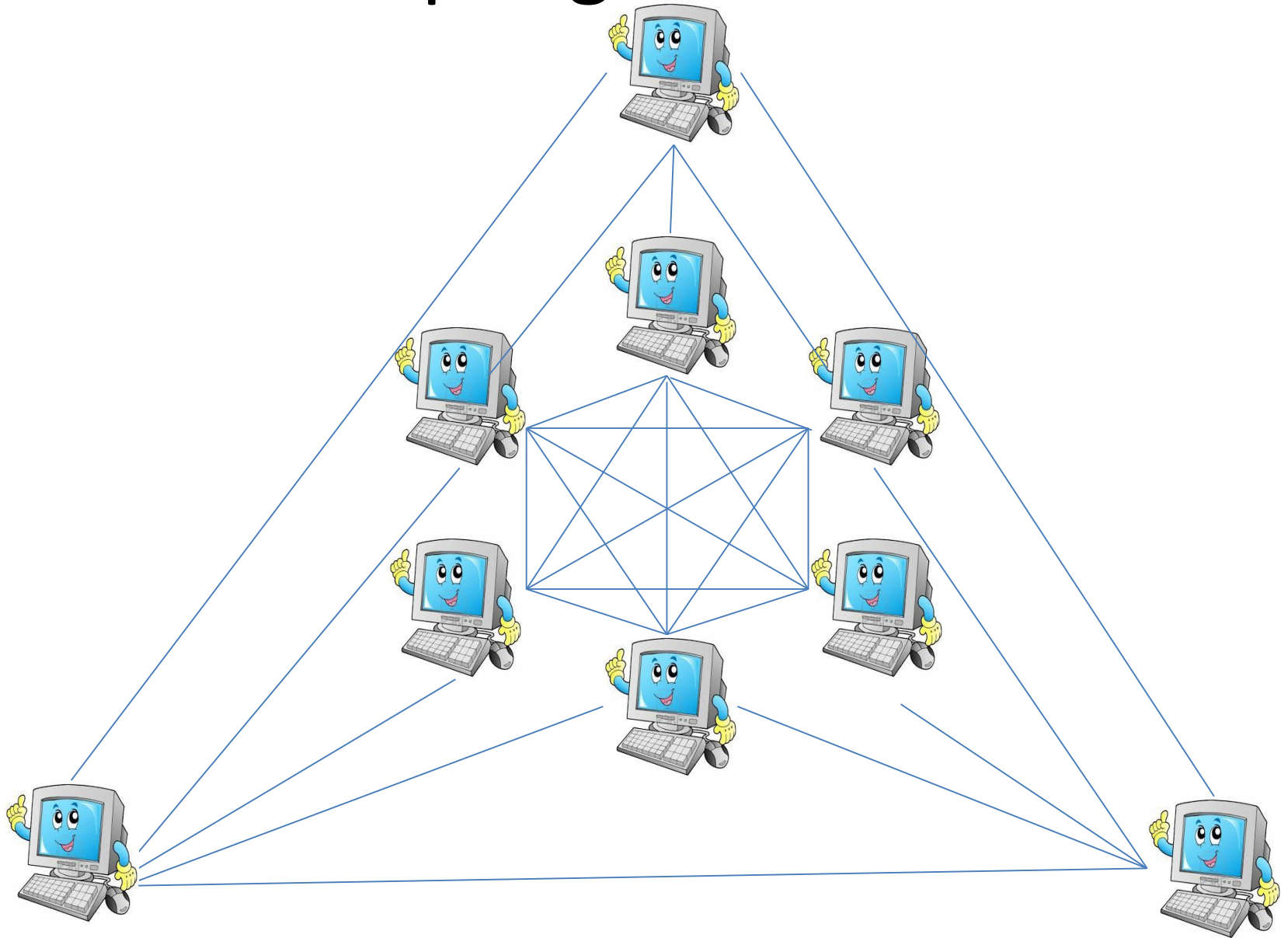
Simulation-Based Security



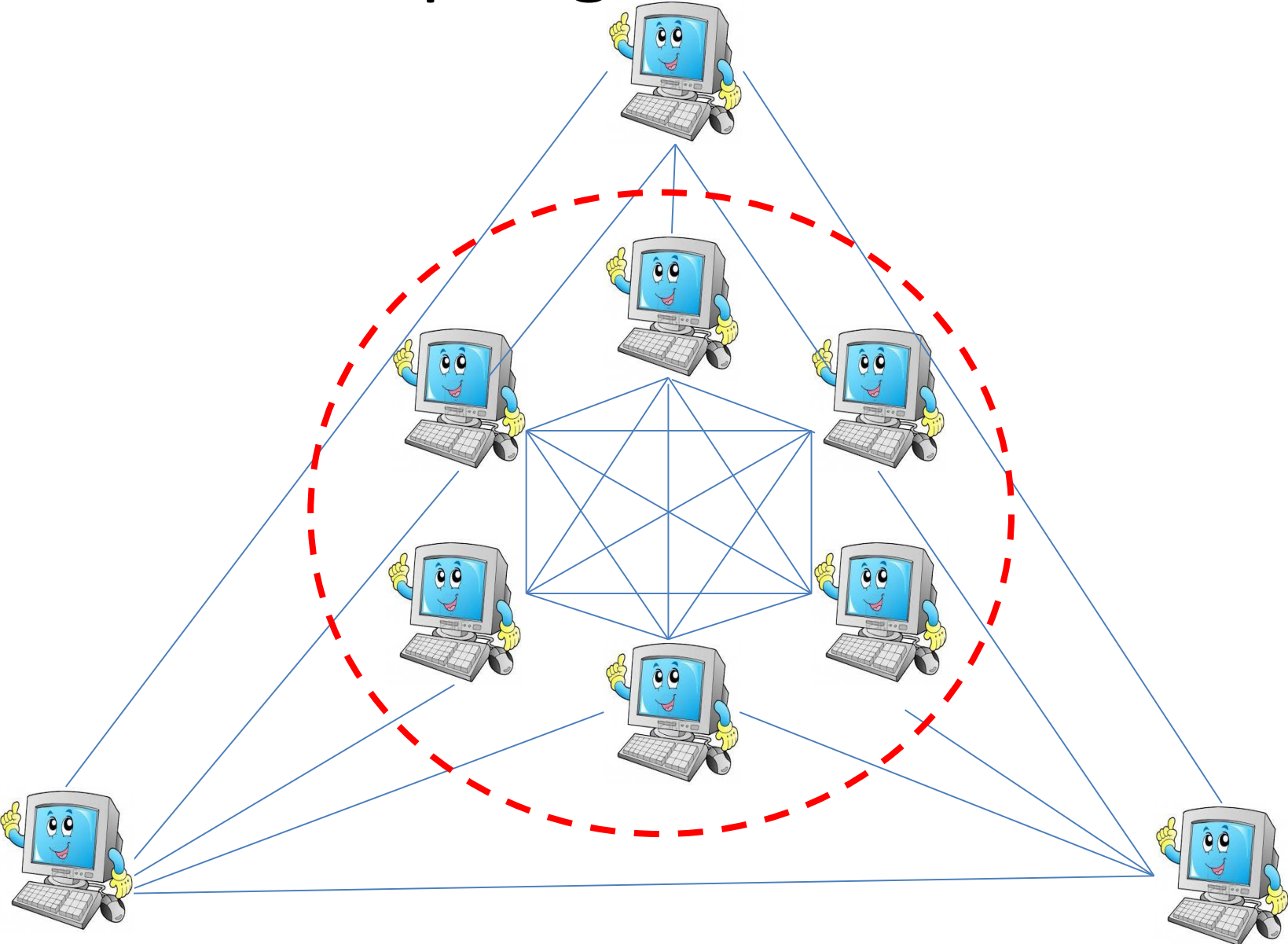
Simulation-Based Security



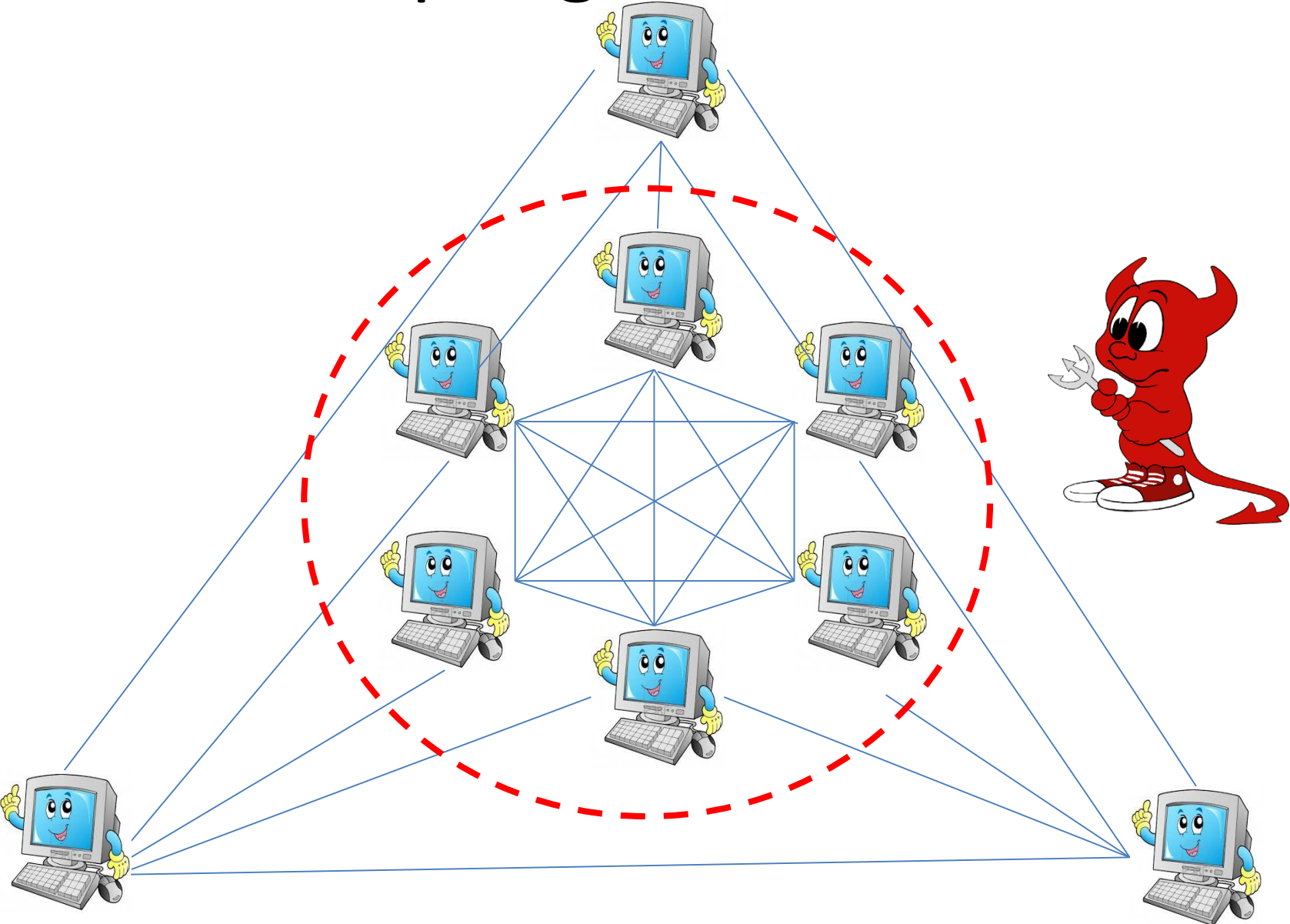
Corrupting All Parties



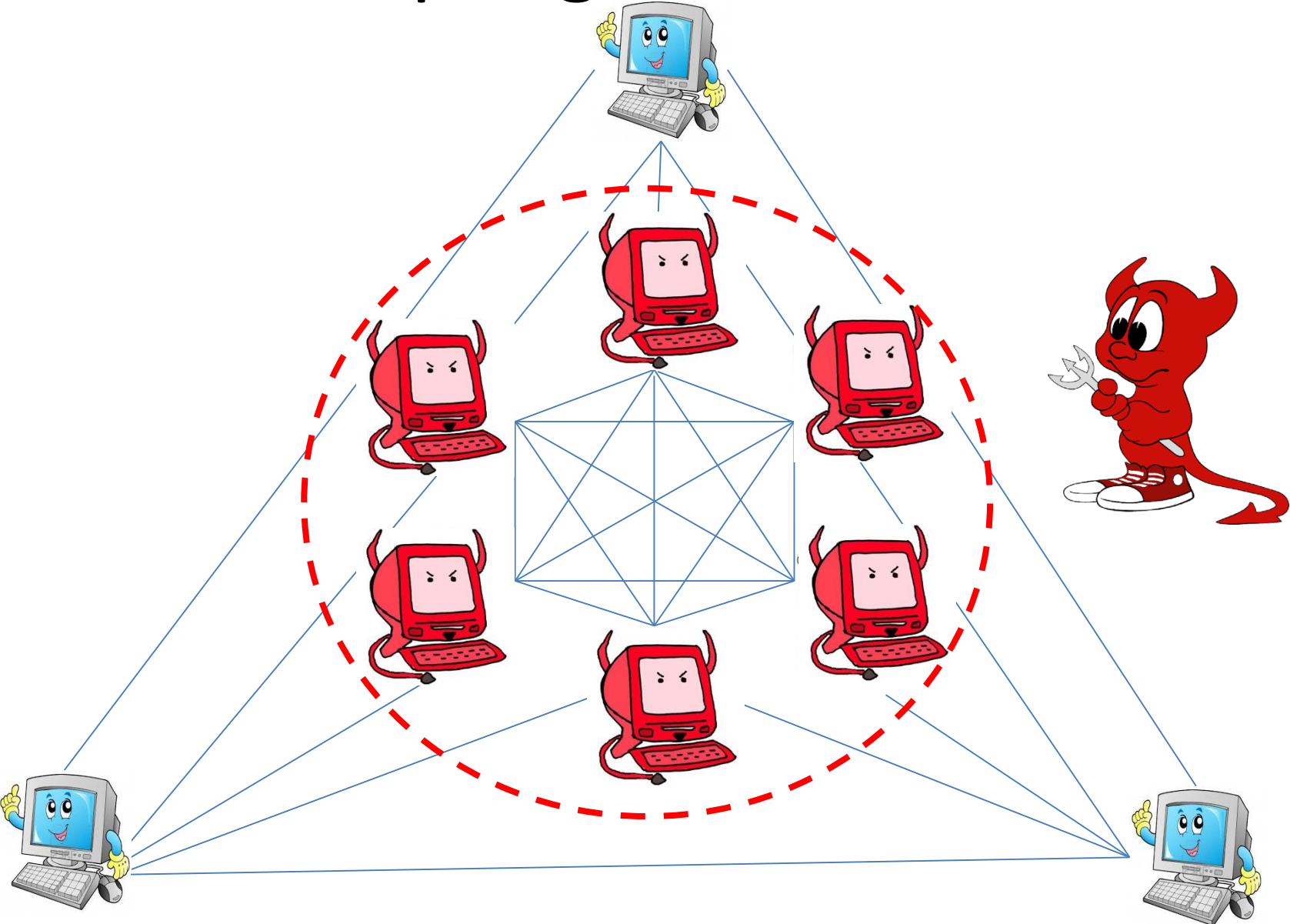
Corrupting All Parties



Corrupting All Parties



Corrupting All Parties



Modeling Adaptive Security

Modular Composition [Canetti'00]	Universal Composition [Canetti'01]
Sequential composition	Concurrent composition
Synchronous protocols	Asynchronous protocols
(Mostly) non-interactive environment	Interactive environment
Inputs are given statically before the computation	Inputs are given dynamically during the computation

Feasibility Result [CLOS'02]

1. Semi-honest protocol in the plain model

– Round complexity is $O(d)$

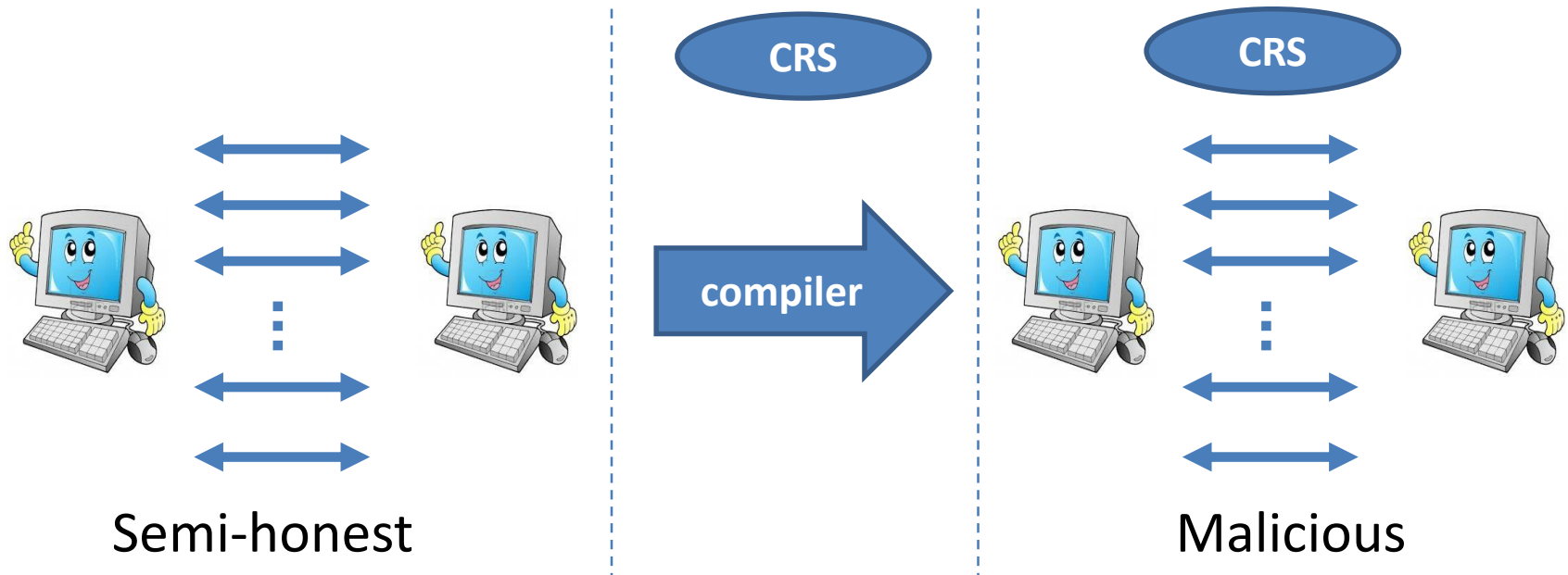
d = depth of the circuit

2. Semi-honest to malicious compiler in CRS model

– Round complexity blows up by constant factor

3. Malicious protocol in CRS model

– Round complexity is $O(d)$



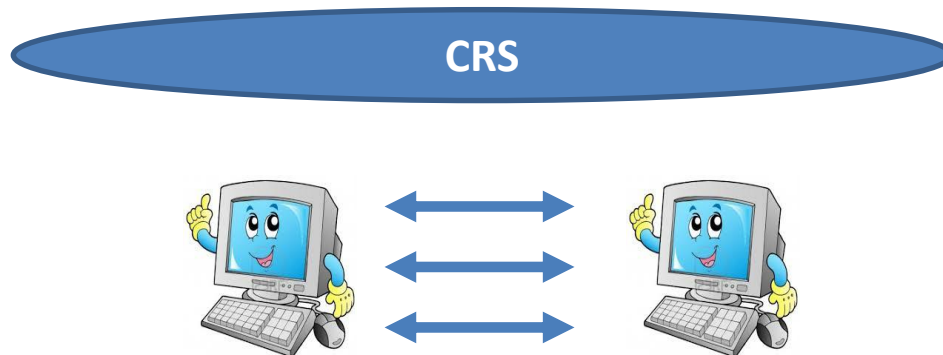
Constant-Round Protocols

Constant-round adaptive MPC [CGP'15] [DKR'15] [GP'15]

- In the CRS model, also for the **semi-honest** case
- CRS contains obfuscated program that gets the circuit as input
⇒ The **size of the CRS** grows with the **size of the circuit**

Constant-round in RAM model [CP'16]

- The **size of the CRS** grows with the **size of the inputs**



Protocols with Short CRS

Semi-honest setting

- No CRS (plain model)

Malicious setting

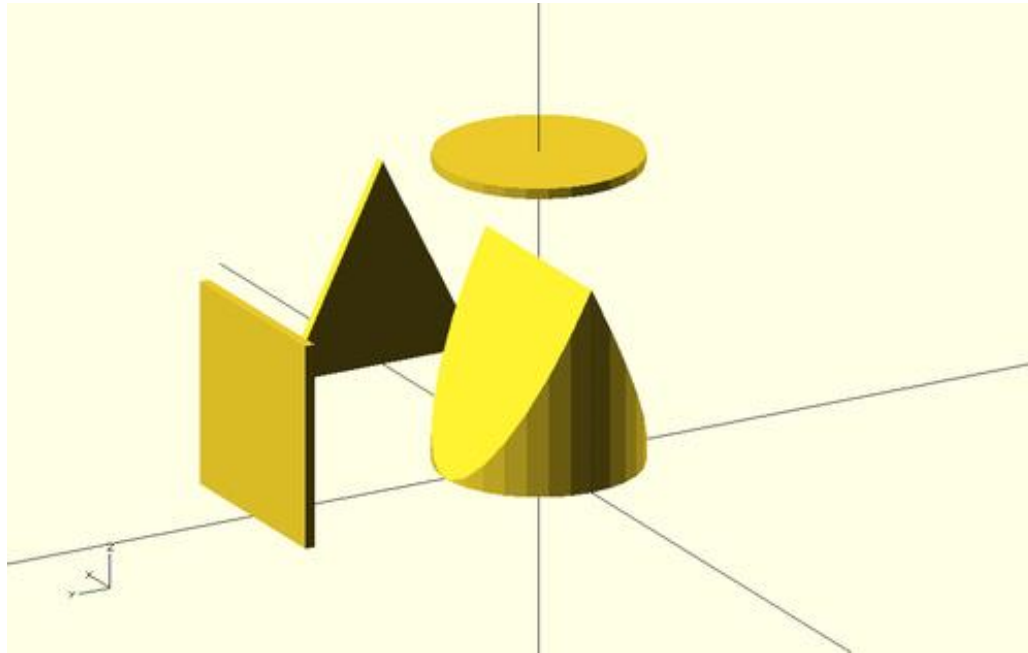
- CRS independent of the circuit
(depends only on security parameter)

Can use [\[CLOS'02\]](#) compiler

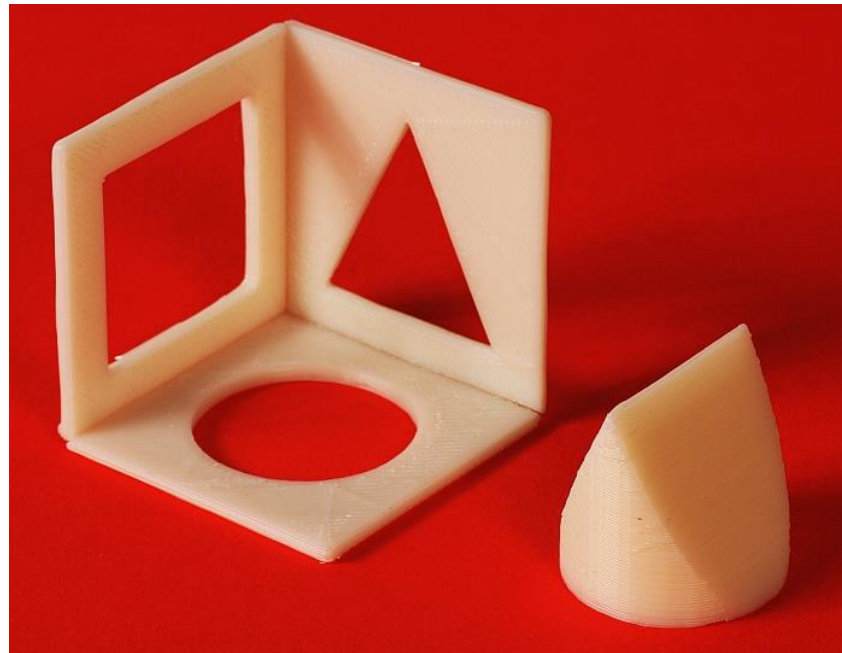
Outline

1. Non-Interactive NCE in UC framework
2. Protocols with round complexity independent of circuit
3. Constant-round protocols for class of functions

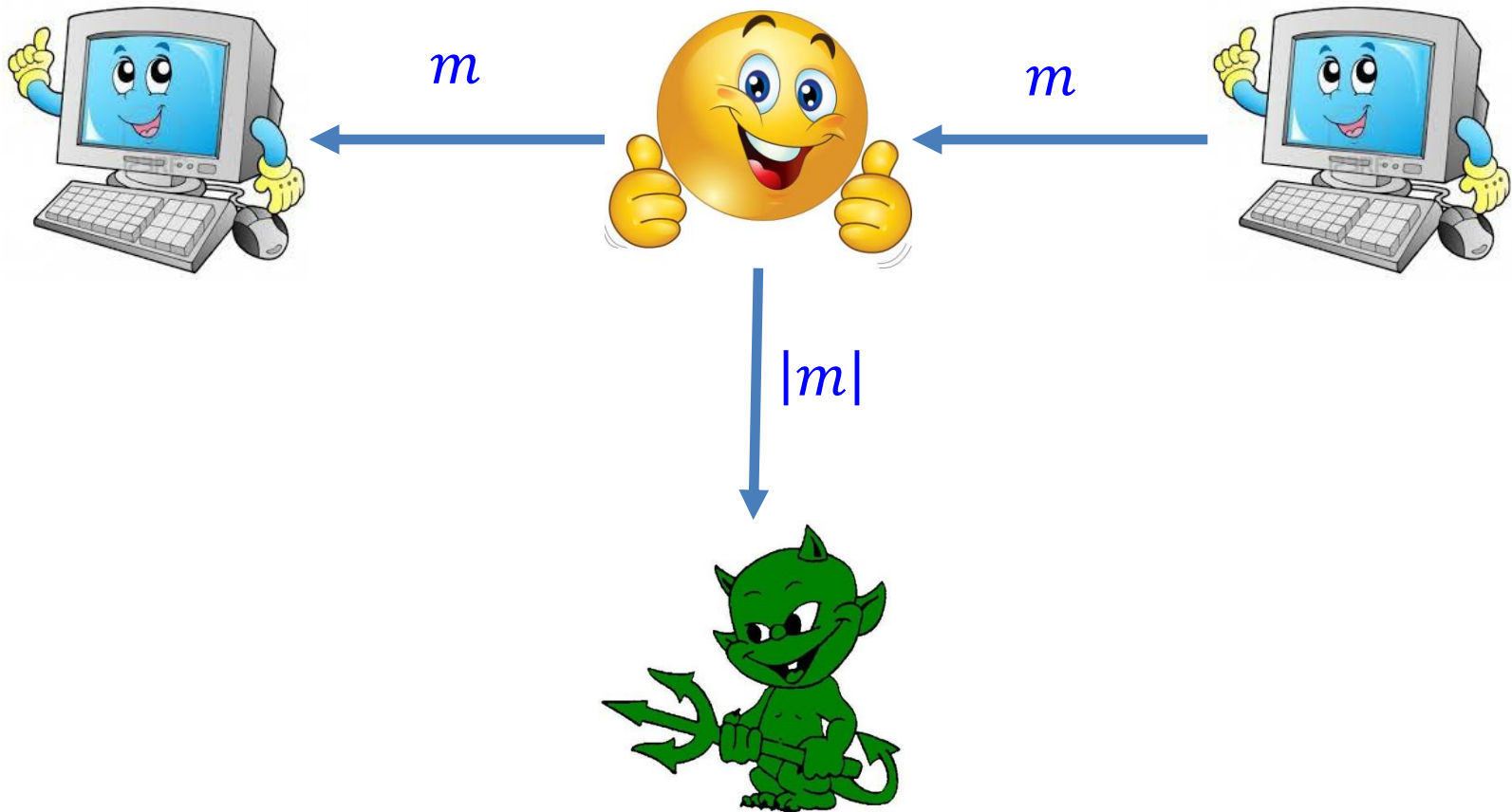
Non-Interactive Non-Committing Encryption



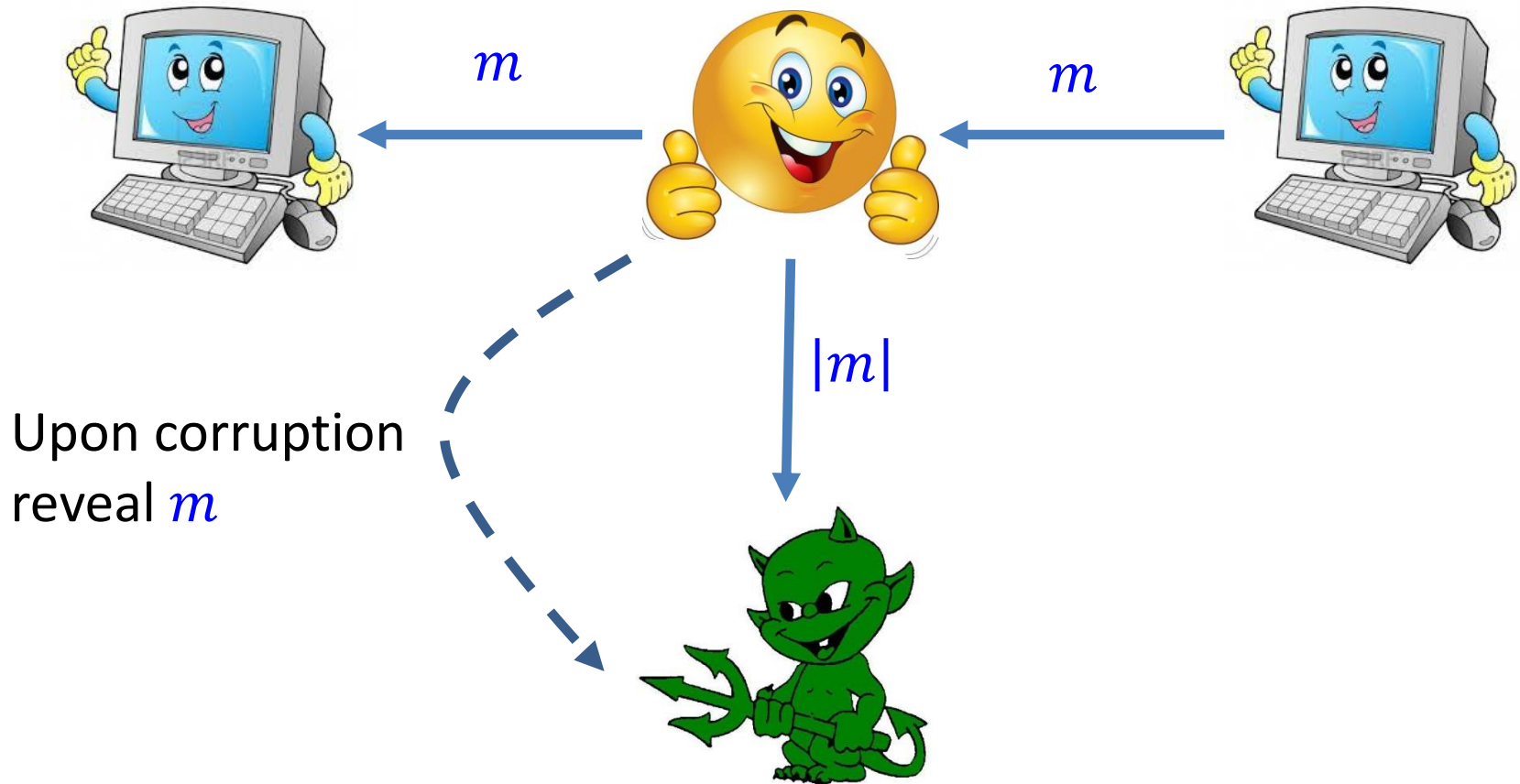
Non-Interactive Non-Committing Encryption



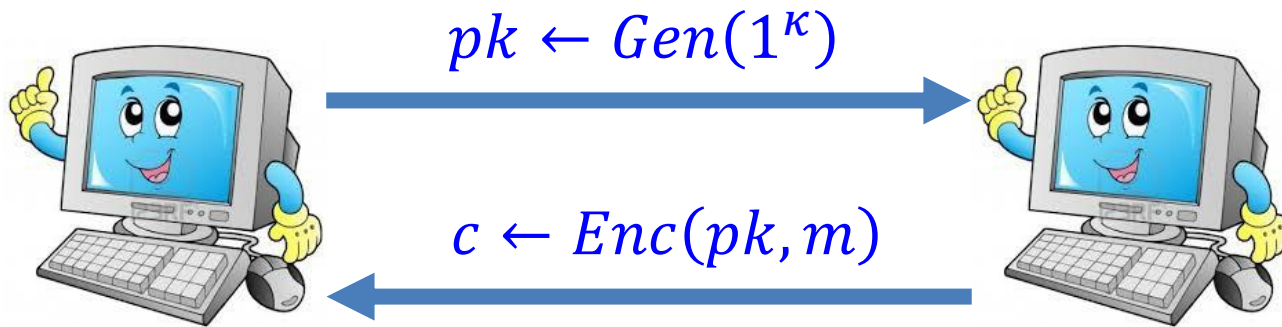
Secure Message Transmission (SMT)



Secure Message Transmission (SMT)



Statically Secure Protocol



- Use public-key encryption (PKE)
- Simulation:
 - Both parties are honest, encrypt 0
 - One party corrupted, \mathcal{S} learns m and encrypts m



PKE can be defined as a non-interactive (2-round) protocol **statically** realizes \mathcal{F}_{SMT}

Adaptive Corruptions

- Using PKE simulation fails when parties start honest
- [CFGN'96] defined **Non-Committing Encryption** (NCE) as n -party protocol that **adaptively** realizes \mathcal{F}_{SMT}
- [DN'00] defined **strong NCE** as 2-party protocol that **adaptively** realizes \mathcal{F}_{SMT} (in [Canetti'00])
- Both definitions and constructions are **interactive**
- Can define **non-interactive** NCE as 2-round protocol
- [CLOS'02] provided a simpler definition

Non-Interactive NCE

Definition: A PKE scheme (Gen, Enc, Dec) with algorithm Sim is non-interactive NCE if $\forall m \in \{0,1\}$ the distributions are comp. indistinguishable

- Honest view of encryption of m

$$\{pk, c, r_G, r_E \mid pk = Gen(1^\kappa; r_G), c = Enc(pk, m; r_E)\}$$

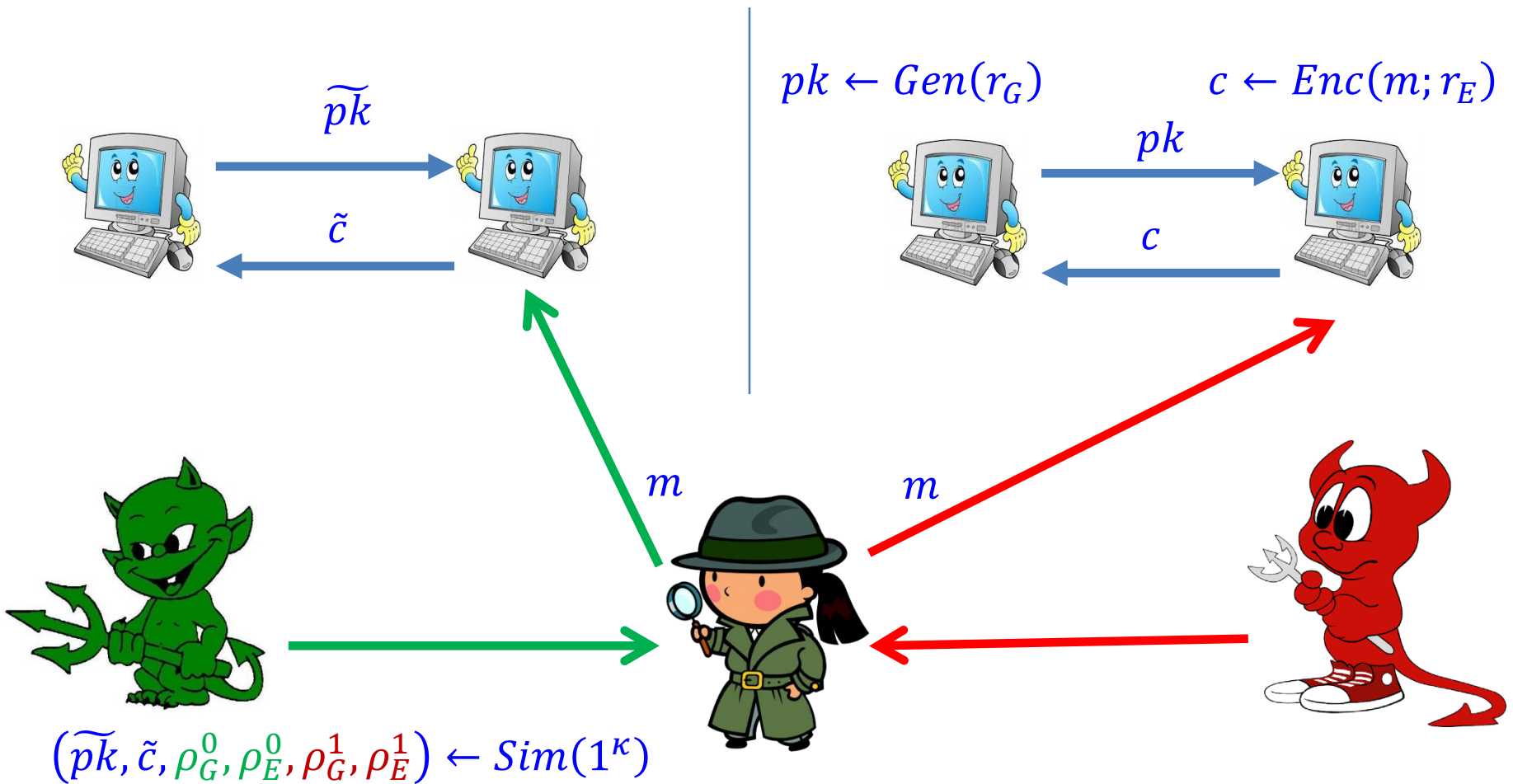
- Simulated encryption explained for m

$$\{\widetilde{pk}, \tilde{c}, \rho_G^m, \rho_E^m \mid (\widetilde{pk}, \tilde{c}, \rho_G^0, \rho_E^0, \rho_G^1, \rho_E^1) \leftarrow Sim(1^\kappa)\}$$

Non-Interactive NCE (2)

IDEAL

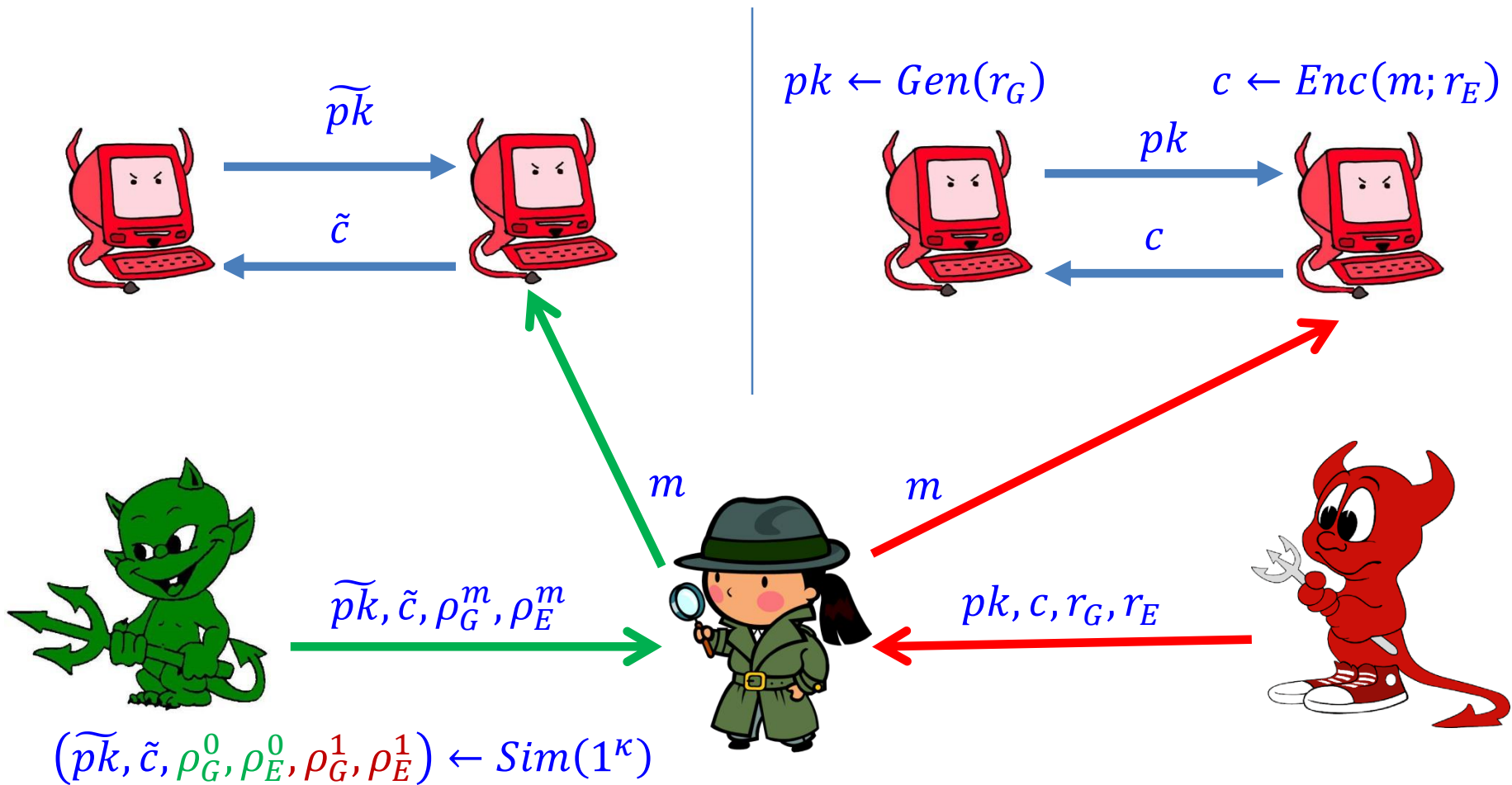
REAL



Non-Interactive NCE (2)

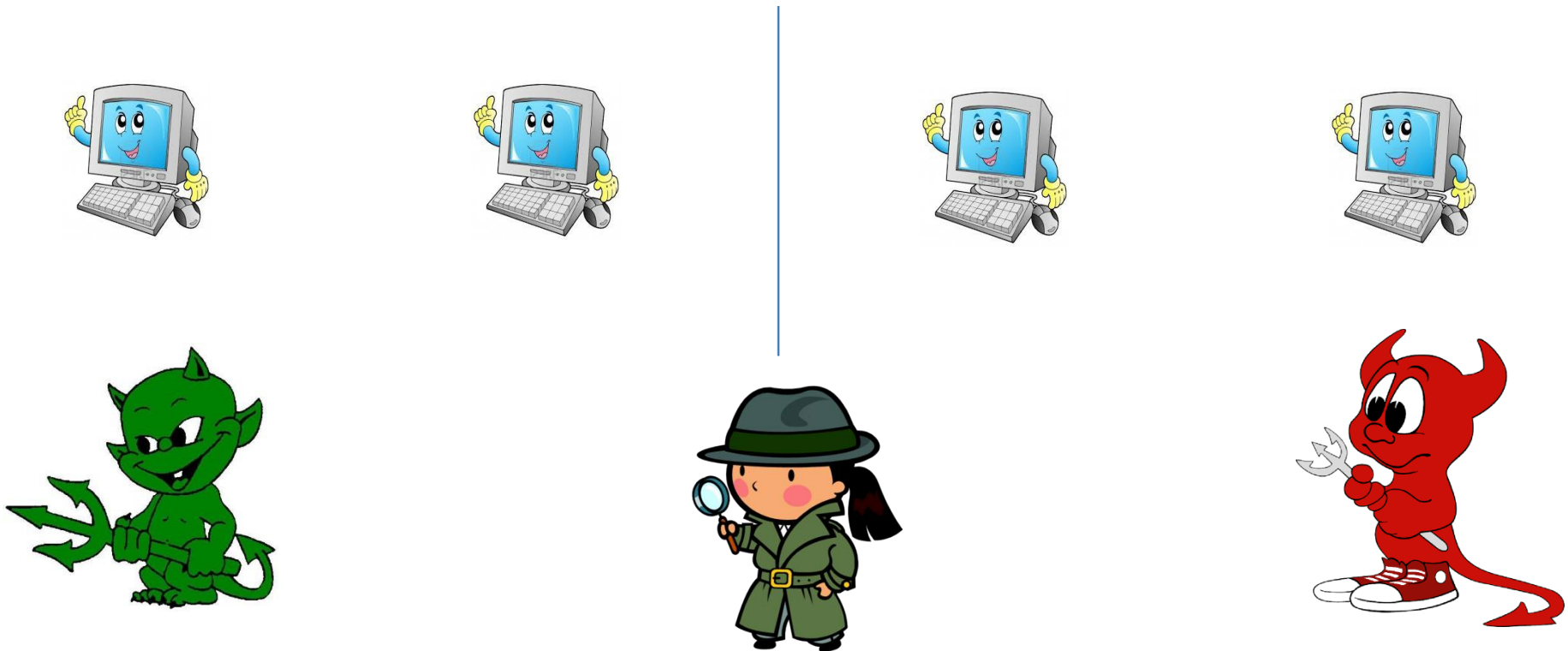
IDEAL

REAL



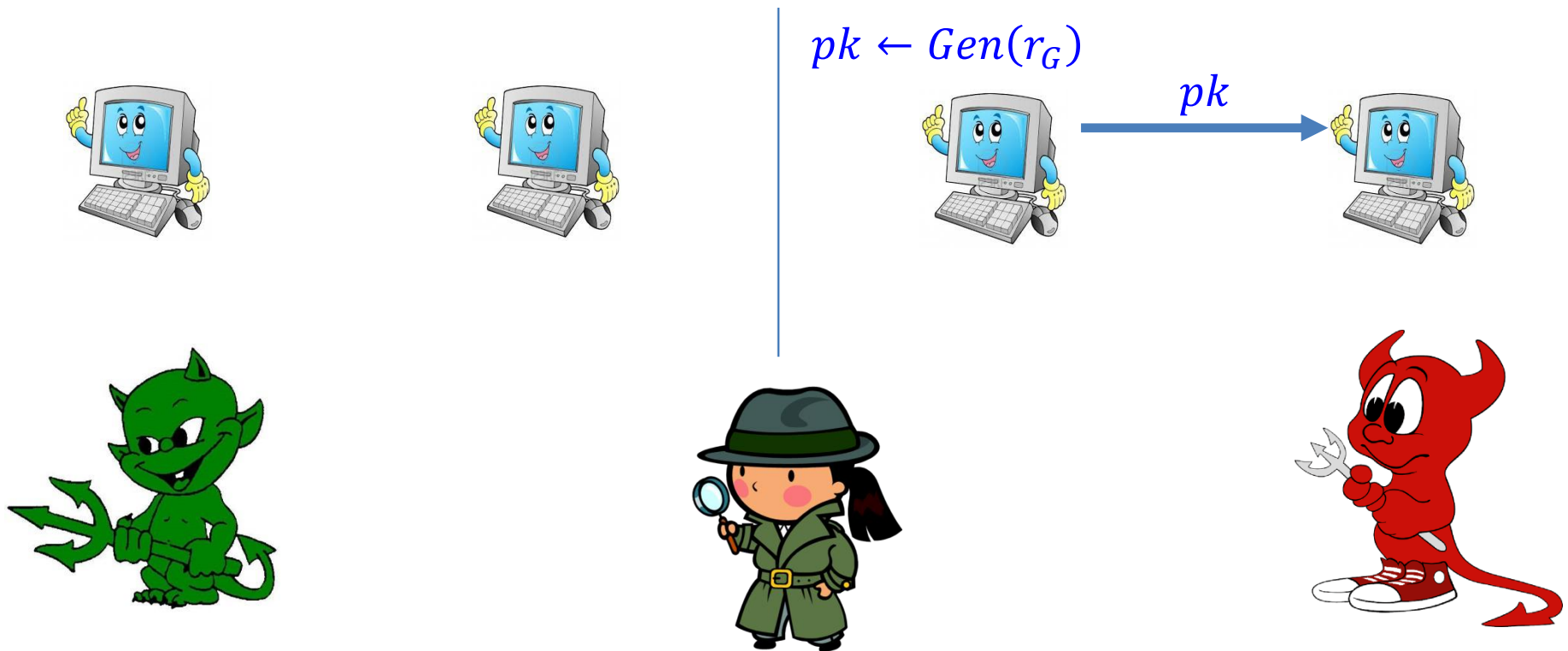
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



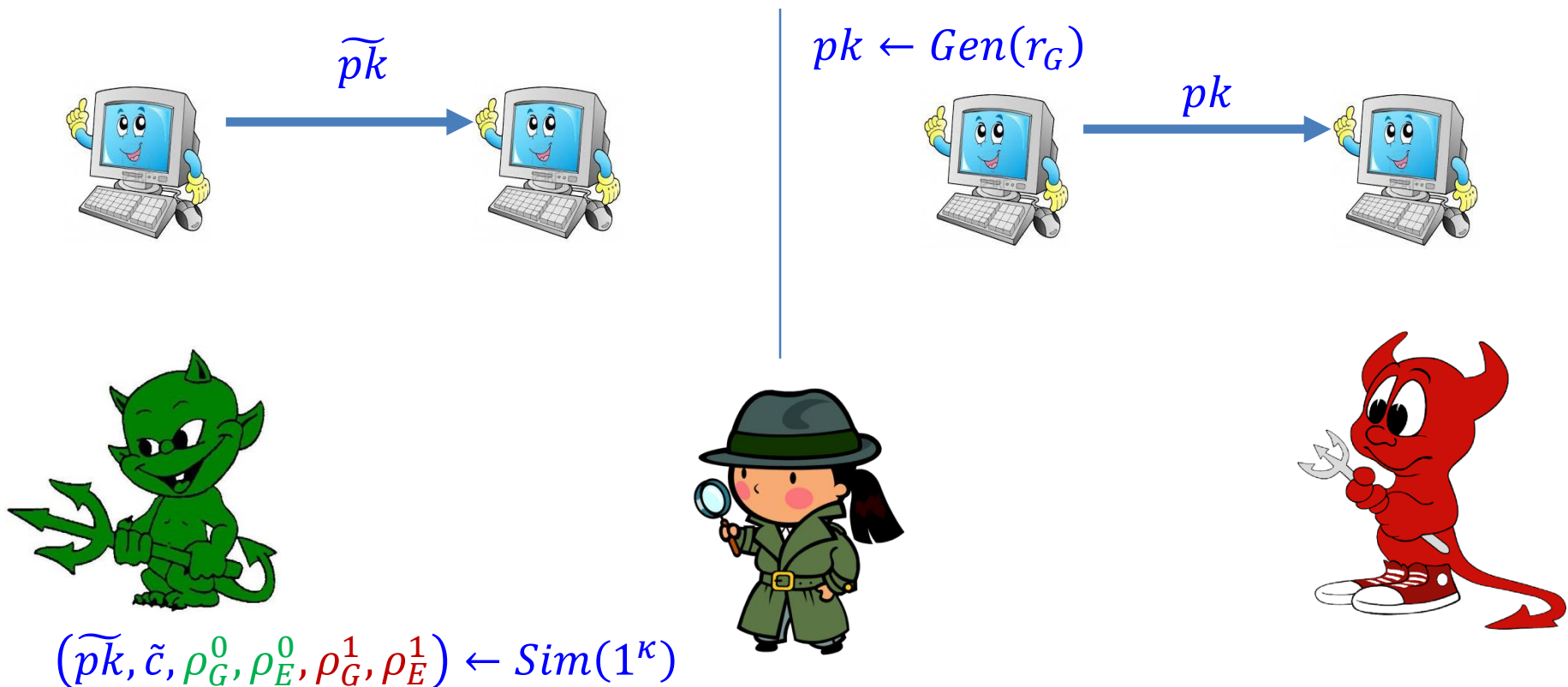
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



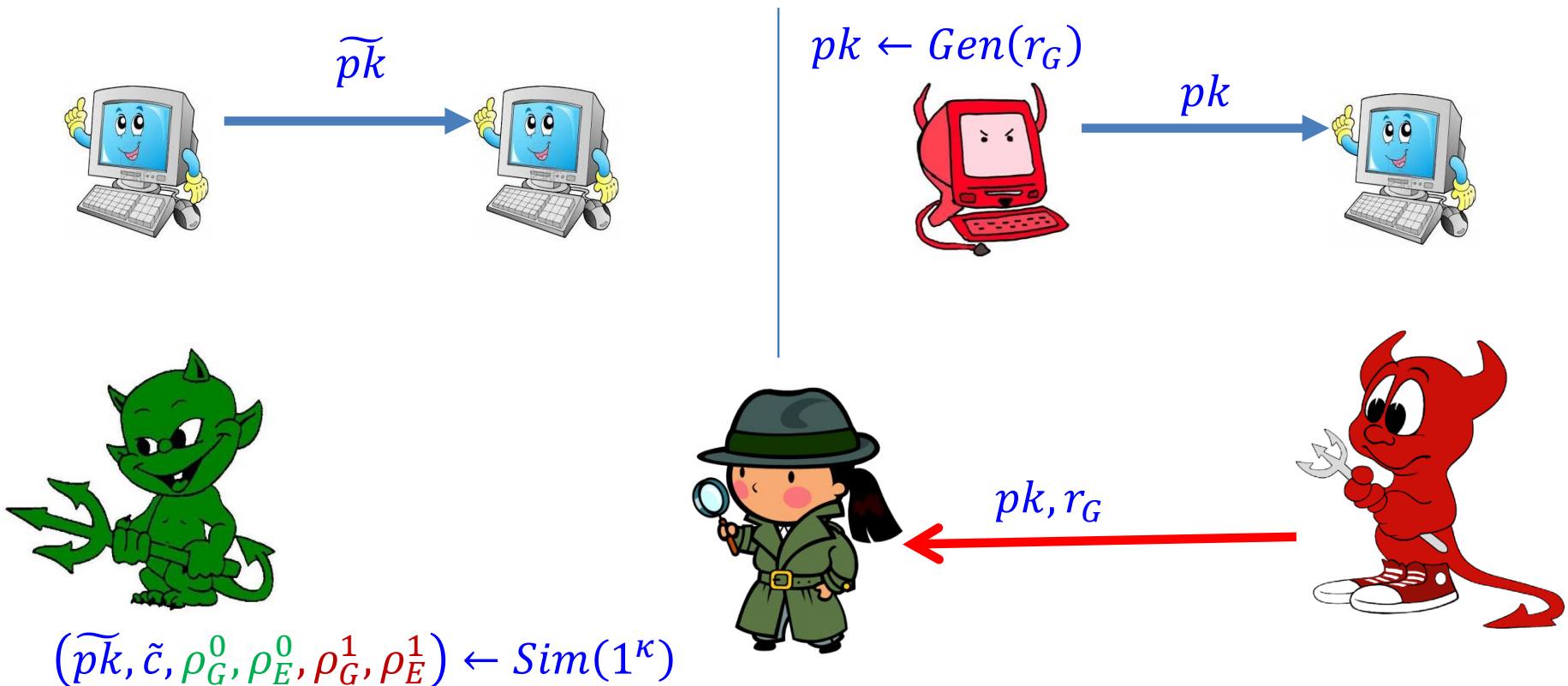
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



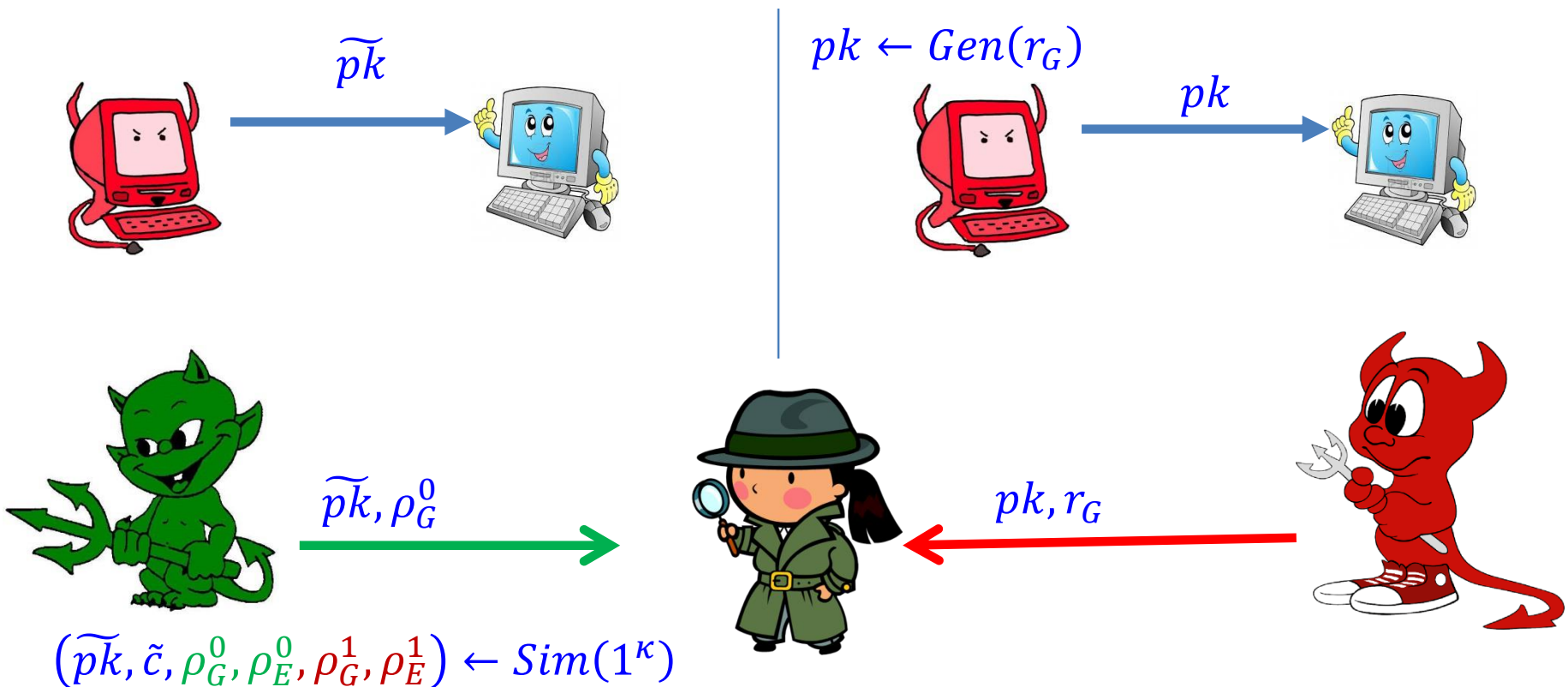
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



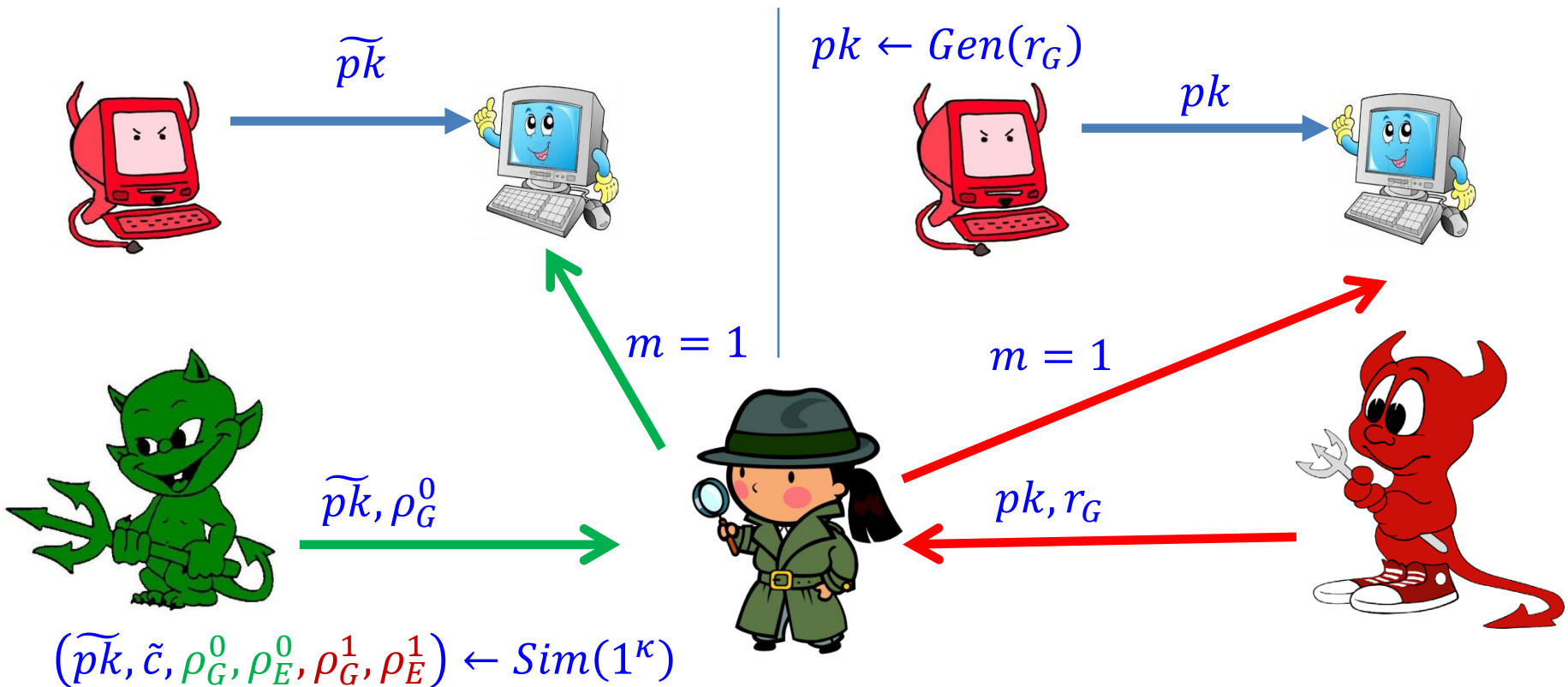
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



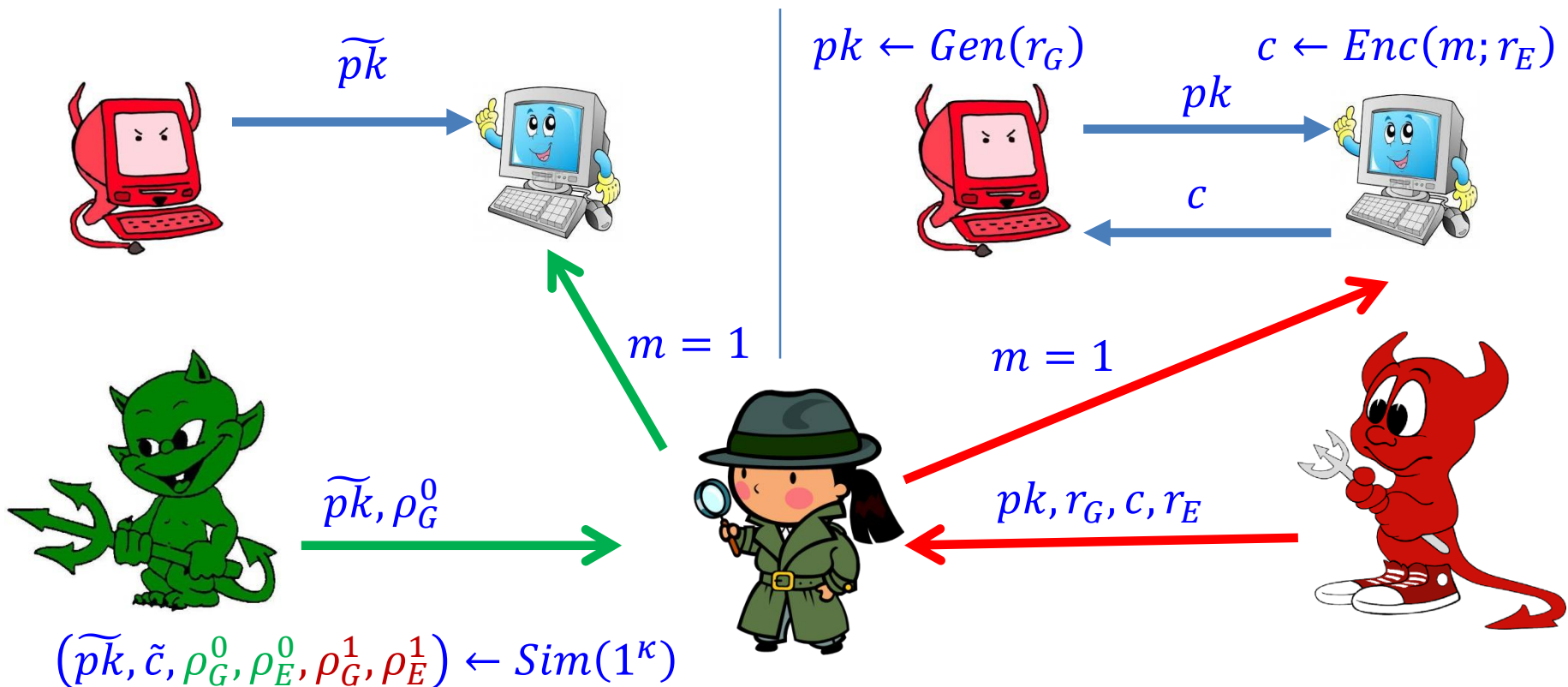
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



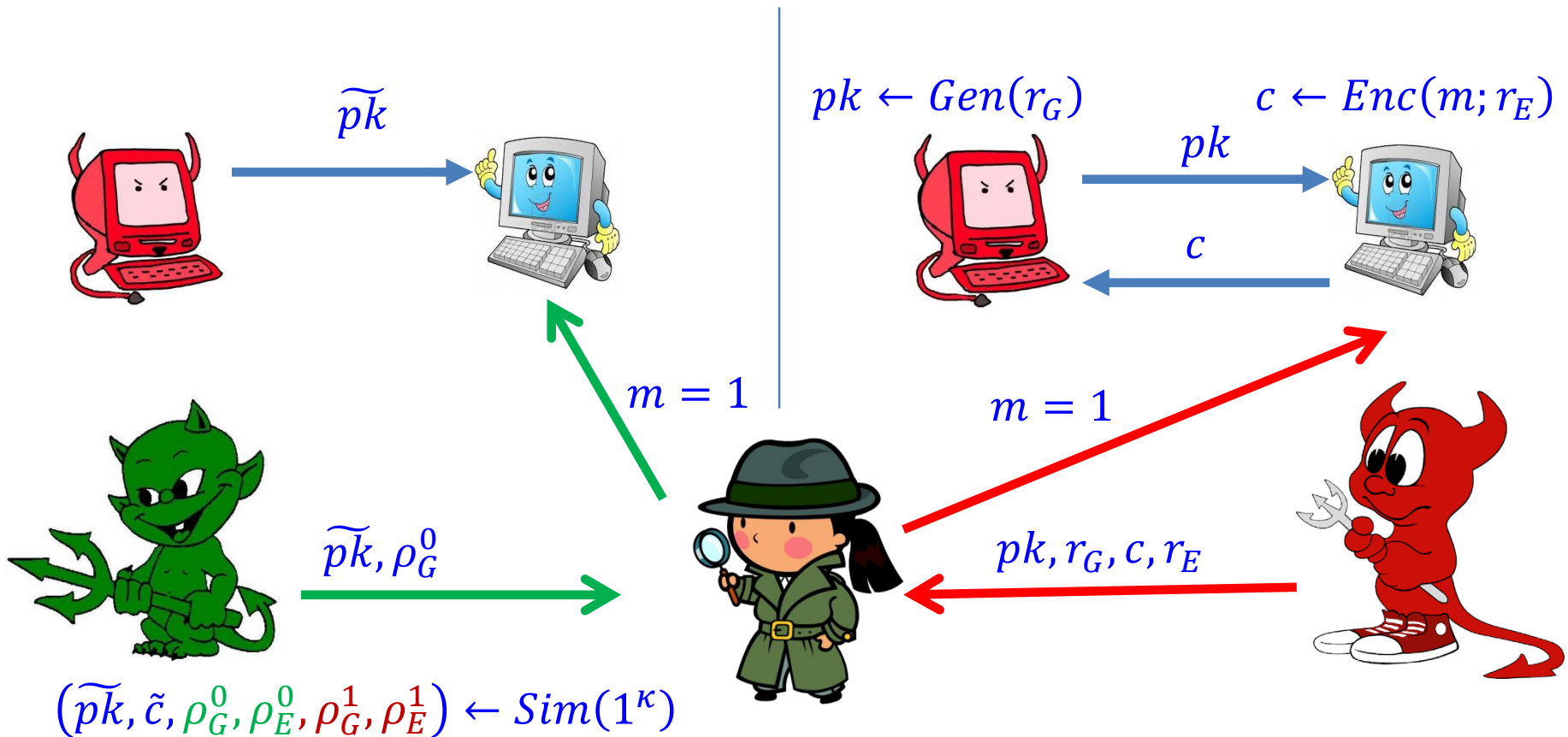
Problem

- Simulation is valid if inputs are given **before** the computation begins (as in modular composition)
- In UC inputs are **dynamically** generated
- Need to simulate corruptions before inputs are given



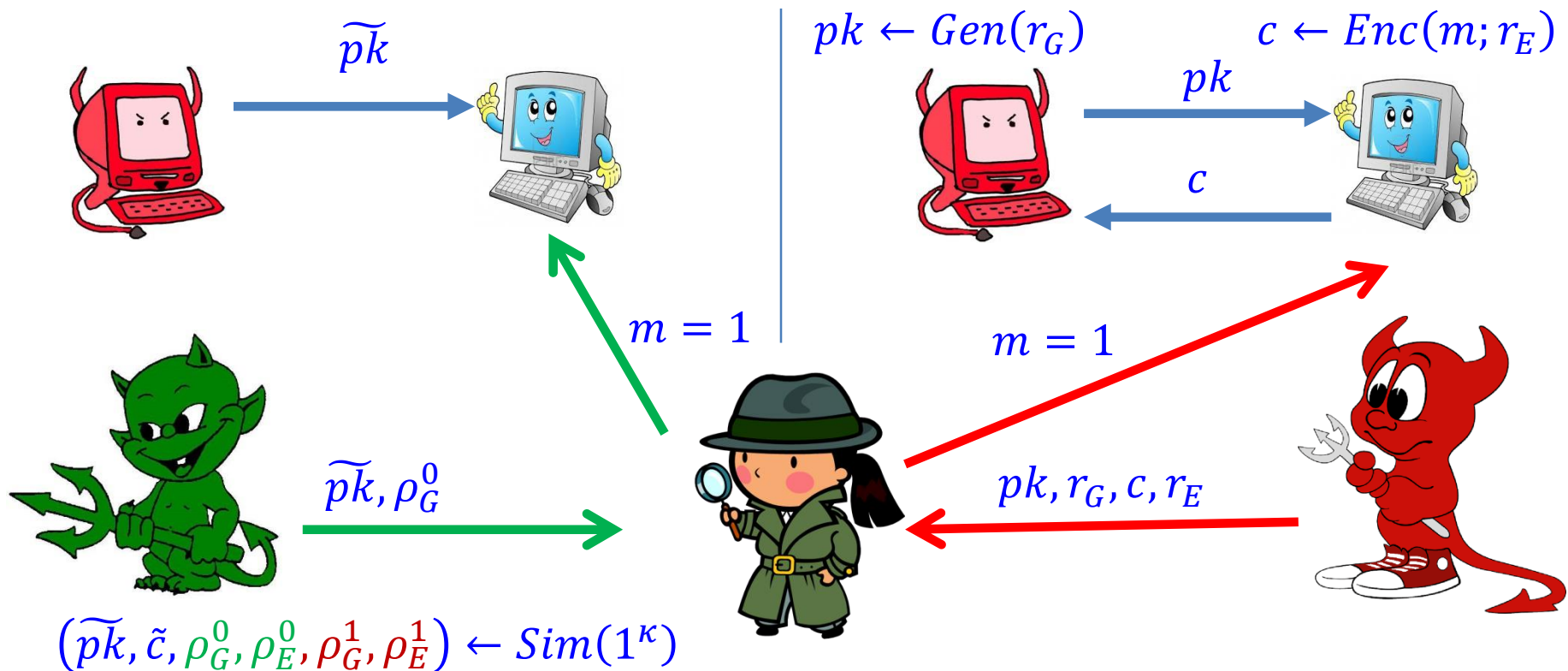
Problem

- Once \widetilde{pk}, ρ_G^0 (or ρ_G^1) are fixed, \tilde{c} is committing
- \tilde{c} won't decrypt to random m with noticeable prob.



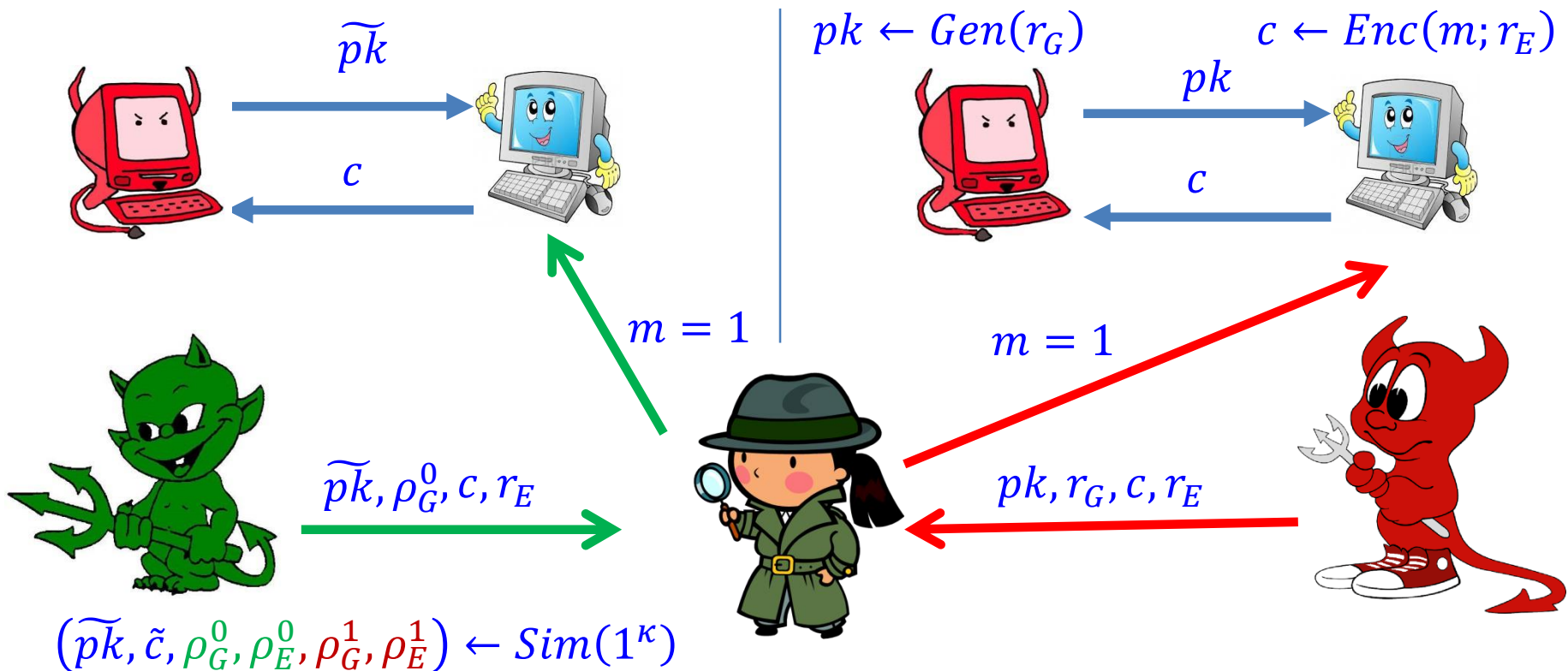
Adjust the Simulation

- Simulation of c only after sender activated with m
- \mathcal{S} learns m from ideal functionality (receiver corrupt)
- \mathcal{S} encrypts $c \leftarrow \text{Enc}(\tilde{pk}, m; r_E)$



Adjust the Simulation

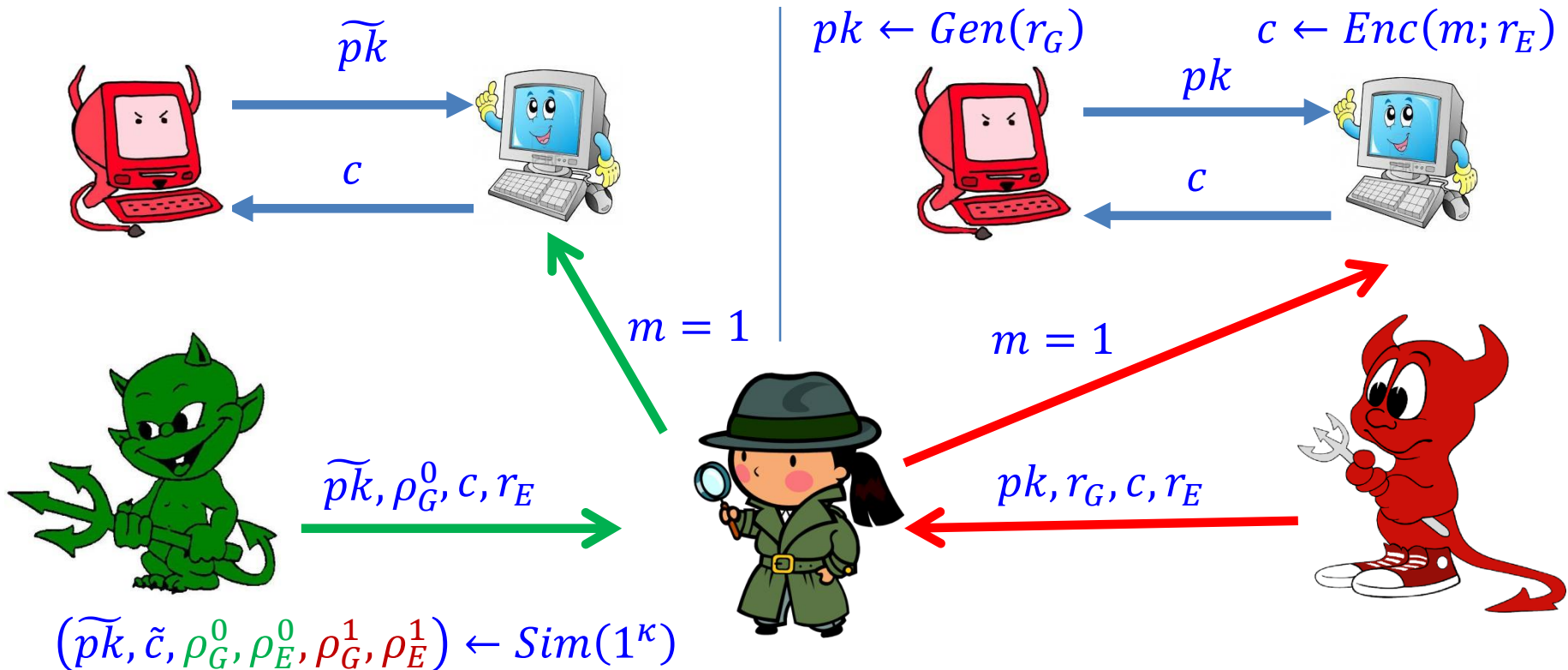
- Simulation of c only after sender activated with m
- \mathcal{S} learns m from ideal functionality (receiver corrupt)
- \mathcal{S} encrypts $c \leftarrow \text{Enc}(\tilde{pk}, m; r_E)$



Adjust the Simulation (2)

- We show how to combine **committing** and **non-committing** ciphertexts in simulation

Thm: If non-interactive NCE exists, then \mathcal{F}_{SMT} can be adaptively UC realized in 2 rounds



Application: Oblivious Transfer (OT)

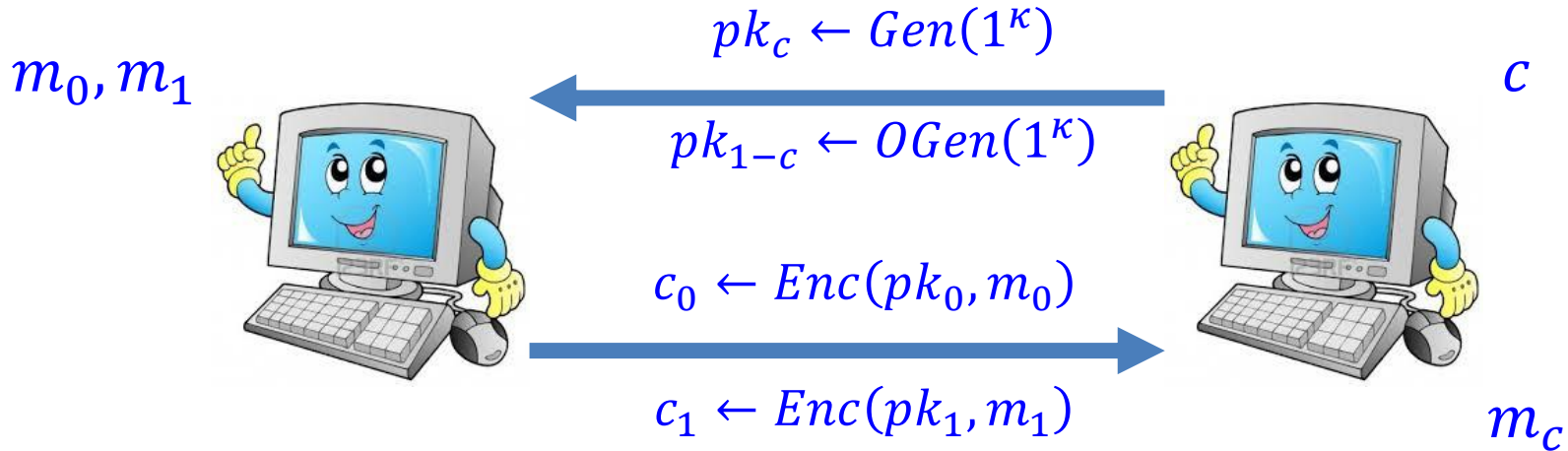


Augmented NCE:

- Oblivious sampling of public keys $pk \leftarrow OGen(1^\kappa)$
- Invertible sampling

$$\{pk, r \mid pk = OGen(1^\kappa; r)\} \sim \{pk, OGen^{-1}(pk) \mid pk \leftarrow Gen(1^\kappa)\}$$

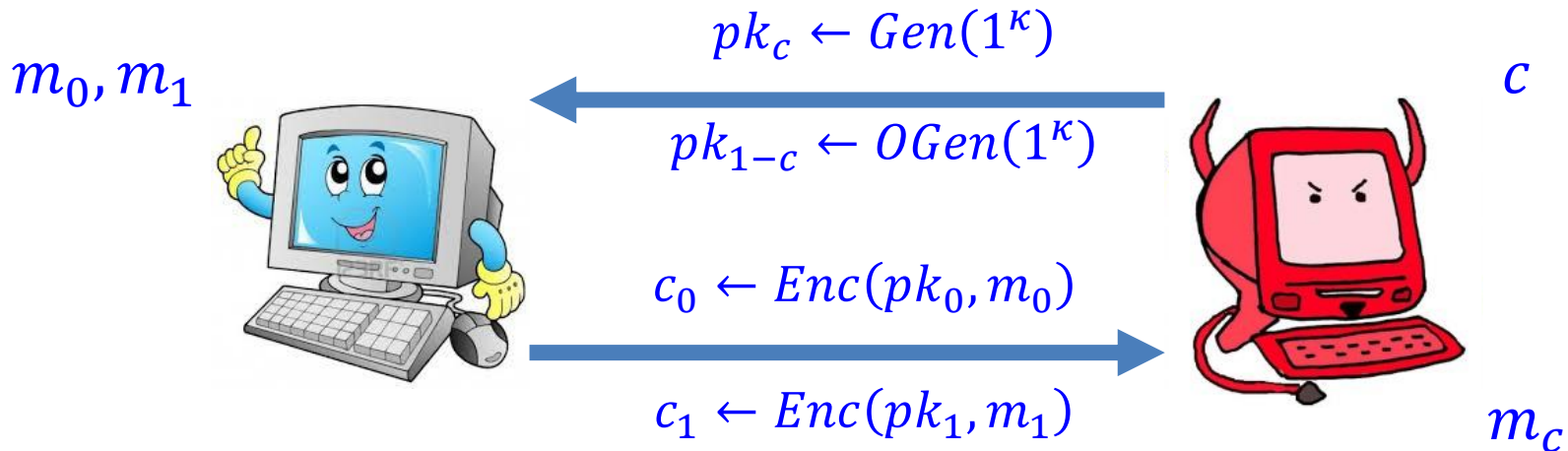
Adaptive OT [CLOS'02]



Simulation (semi-honest)

- \mathcal{S} simulate using $(\widetilde{pk}_0, \tilde{c}_0, \rho_{0,G}^0, \rho_{0,E}^0, \rho_{0,G}^1, \rho_{0,E}^1) \leftarrow Sim(1^\kappa)$
 $(\widetilde{pk}_1, \tilde{c}_1, \rho_{1,G}^0, \rho_{1,E}^0, \rho_{1,G}^1, \rho_{1,E}^1) \leftarrow Sim(1^\kappa)$

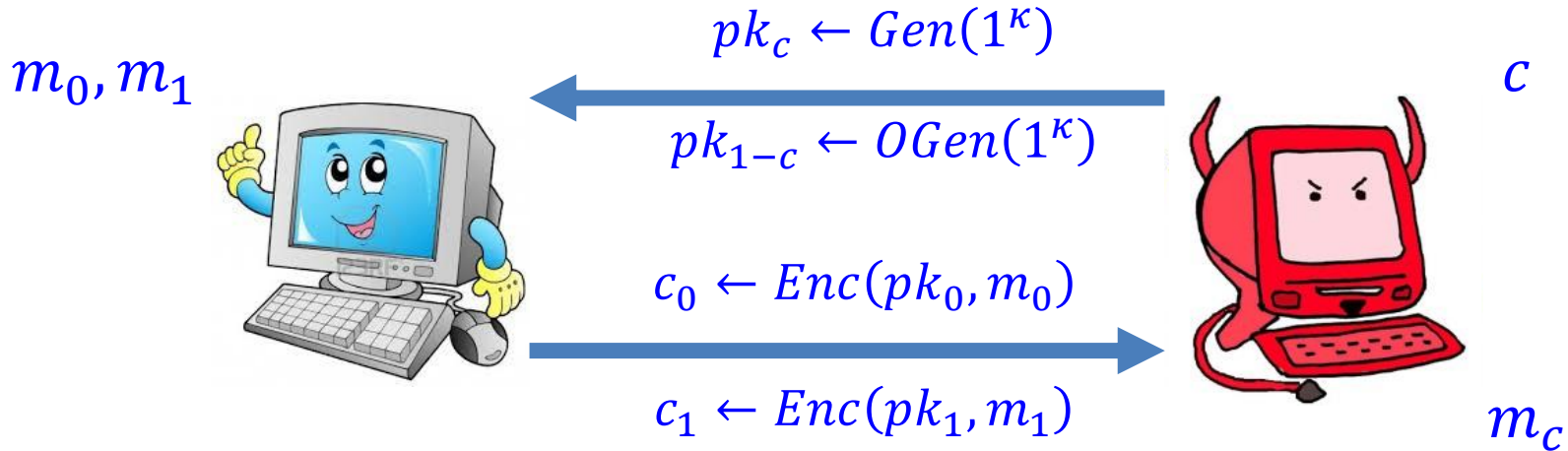
Adaptive OT [CLOS'02]



Simulation (semi-honest)

- \mathcal{S} simulate using $(\widetilde{pk}_0, \tilde{c}_0, \rho_{0,G}^0, \rho_{0,E}^0, \rho_{0,G}^1, \rho_{0,E}^1) \leftarrow Sim(1^\kappa)$
 $(\widetilde{pk}_1, \tilde{c}_1, \rho_{1,G}^0, \rho_{1,E}^0, \rho_{1,G}^1, \rho_{1,E}^1) \leftarrow Sim(1^\kappa)$
- Upon receiver corruption, \mathcal{S} learns c, m_c and provides randomness $\rho_{c,G}^{m_c}$

Adaptive OT [CLOS'02]



Simulation (semi-honest)

- \mathcal{S} simulate using $(\widetilde{pk}_0, \tilde{c}_0, \rho_{0,G}^0, \rho_{0,E}^0, \rho_{0,G}^1, \rho_{0,E}^1) \leftarrow Sim(1^\kappa)$
 $(\widetilde{pk}_1, \tilde{c}_1, \rho_{1,G}^0, \rho_{1,E}^0, \rho_{1,G}^1, \rho_{1,E}^1) \leftarrow Sim(1^\kappa)$
- Upon receiver corruption, \mathcal{S} learns c, m_c and provides randomness $\rho_{c,G}^{m_c}$

See the paper for details

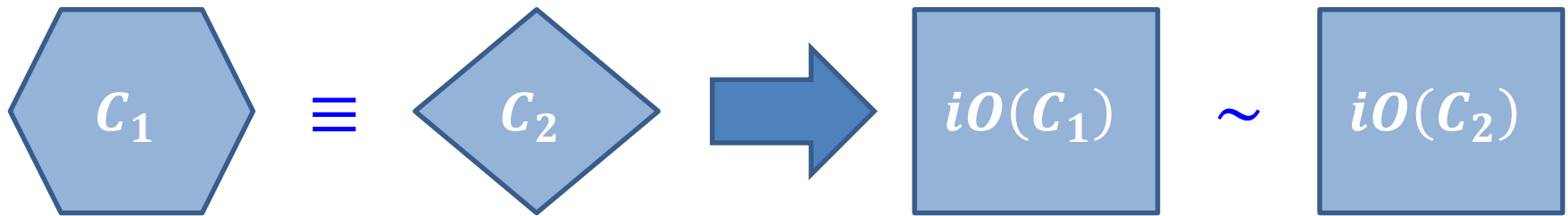
input & output

Round Complexity Independent of C



"Your proposal is written with clarity and conviction. Send it up to legal for obfuscation."

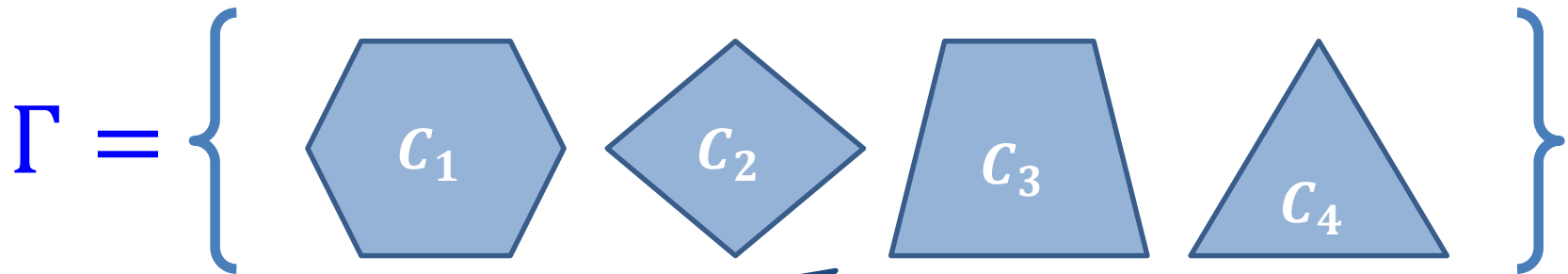
Indistinguishability Obfuscation (iO)



Candidate construction [GGHRSW'13]


Nice property: the **depth** of the **obfuscation circuit** is independent of the **circuit to obfuscate**

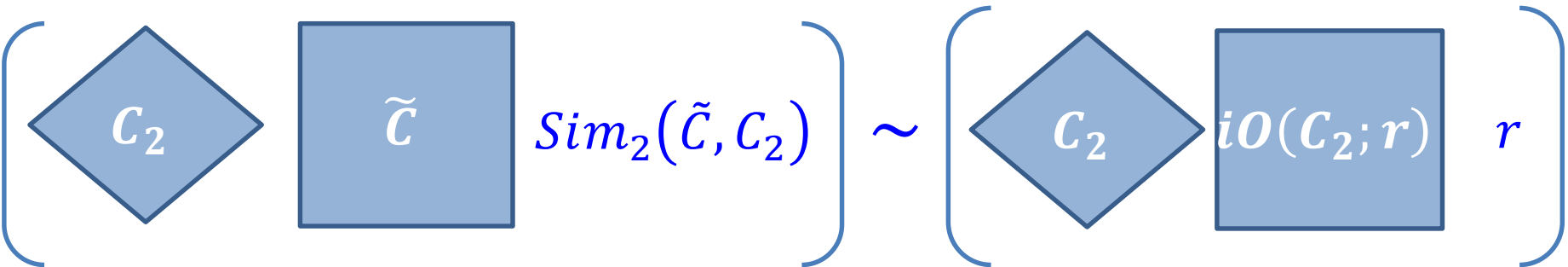
Non-Committing iO



All circuits are functionally equivalent

Def: (iO, Sim_1, Sim_2) is non-committing iO for Γ if

- Sim_1 generates canonical obf. circuit \tilde{C} for Γ 
- Given any $C \in \Gamma$, Sim_2 can explain \tilde{C} as $iO(C)$



Non-Committing iO (2)

Bad news:

If NCiO for circuits exists

⇒ poly-time solution to **circuit equivalence** (co-NP)

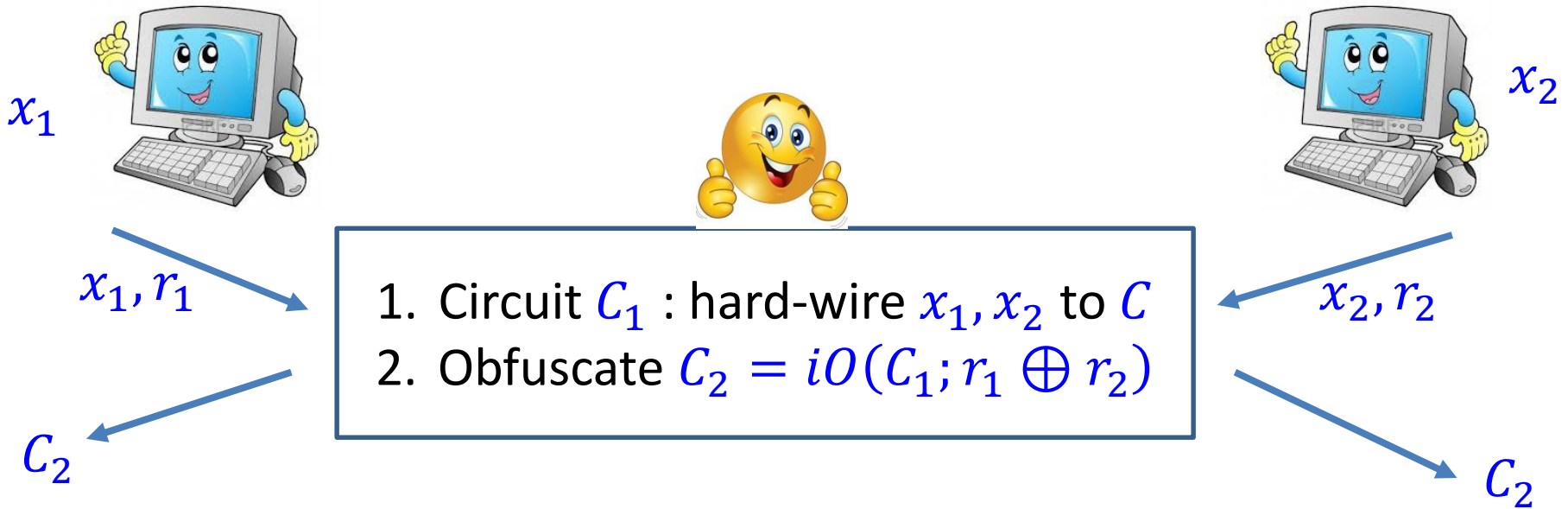
⇒ polynomial hierarchy collapses

Good news:

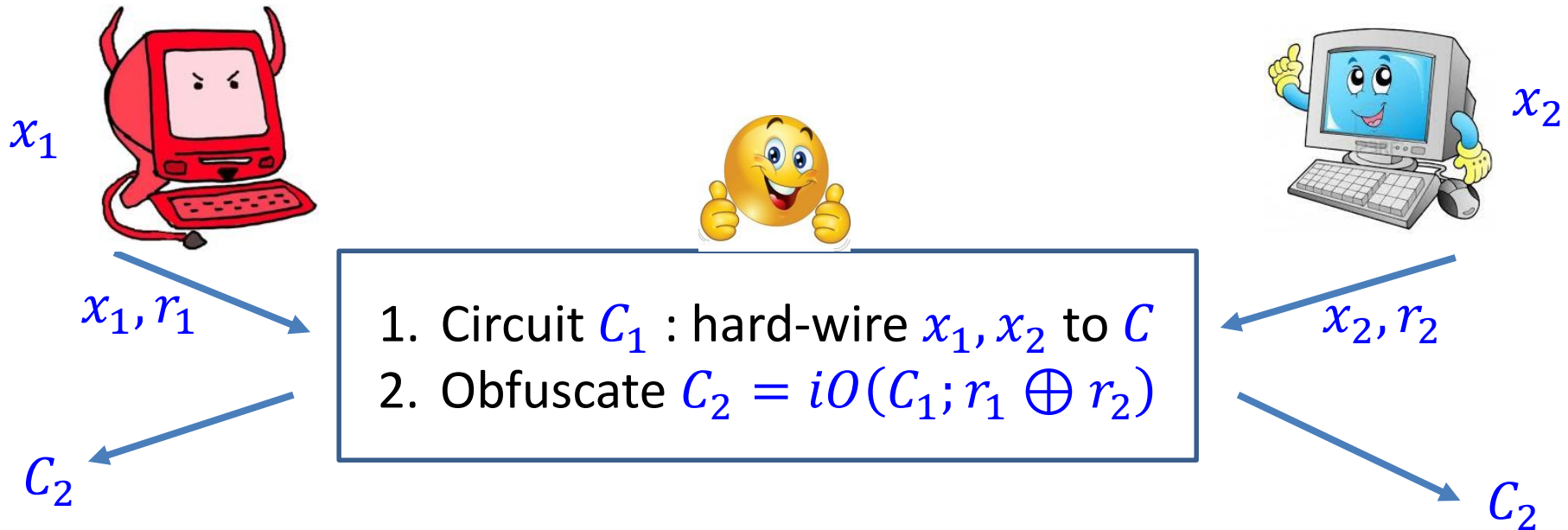
Circuit equivalence is easy for **constant circuits**
(no input wires)

Thm: If NCiO for constant circuits exists
then \exists adaptive SFE protocol with short CRS
whose round complexity is independent of C

Protocol Idea



Protocol Idea

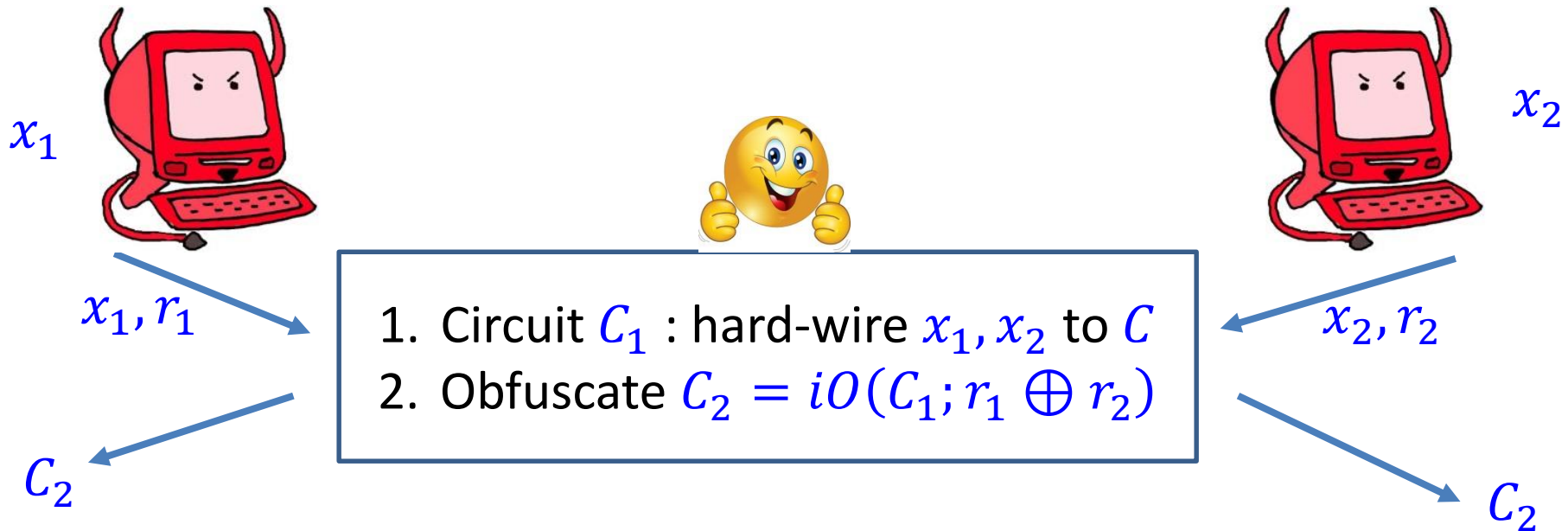


Simulation idea

1st corruption: learn x_1, y and randomly sample r_1

Compute $\tilde{C} \leftarrow Sim_1$ obfuscated constant circuit with output y

Protocol Idea



Simulation idea

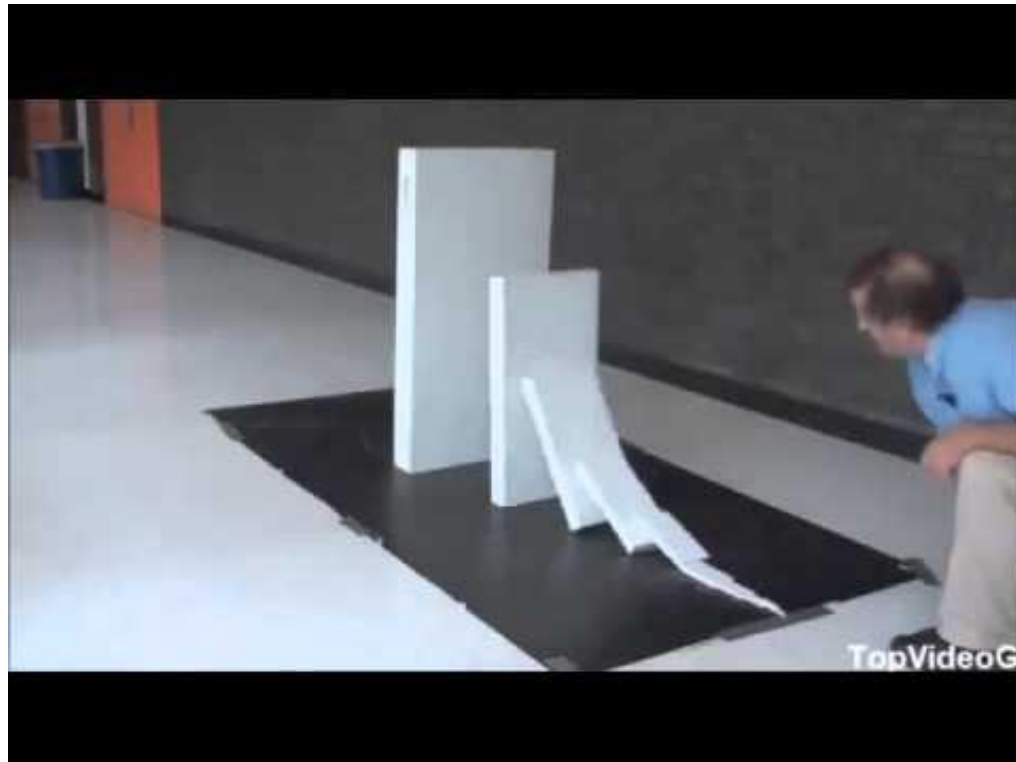
1st corruption: learn x_1, y and randomly sample r_1

Compute $\tilde{C} \leftarrow Sim_1$ obfuscated constant circuit with output y

2nd corruption: learn x_2, y and compute C_1 (using C, x_1, x_2)

Compute $r \leftarrow Sim_2(\tilde{C}, C_1)$ and set $r_2 = r \oplus r_1$

Constant Round for One-Sided Poly-Size Domain



Constant-Round Protocol

Thm: Assume adaptively secure OT exist

- f is deterministic 2-party functionality
- $x_1 \in D \subset \{0,1\}^n$, $|D| = \text{poly}(n)$
- $x_2 \in \{0,1\}^n$

Then f can be adaptively realized with short CRS in constant number of rounds

Optimistic view: feasibility result

Pessimistic view: to rule out constant-round protocols in general, consider **super-poly domain** or **randomized** functions

Summary

1. How to simulate non-interactive NCE in UC
2. NCiO is complete for round complexity ind. of circuit
3. Constant-round protocols for class of functions

Open questions:

- Does NCiO for constant circuits exist?
- Find more functions that have constant-round protocols with short CRS

Grazie

