ACHIEVING PRIVACY IN VERIFIABLE COMPUTATION WITH MULTIPLE SERVERS — WITHOUT FHE AND WITHOUT PREPROCESSING

Prabhanjan Ananth, UCLA
Nishanth Chandran, Microsoft Research India
Vipul Goyal, Microsoft Research India
Bhavana Kanukurthi, UCLA
Rafail Ostrovsky, UCLA

Presented by: Chongwon Cho, HRL

OUTSOURCING COMPUTATION

- Alice has a weak device and she wants to perform expensive computation

OUTSOURCING COMPUTATION

- Alice has a weak device and she wants to perform expensive computation

Solution: Outsource to the cloud

CLOUD COMPUTING

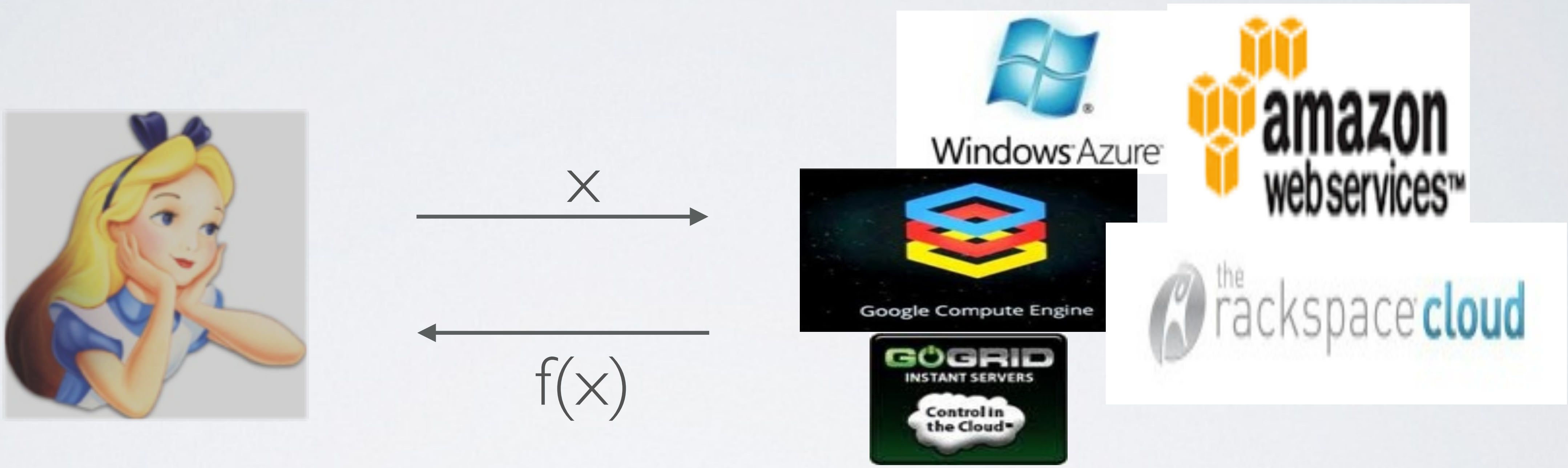
- Commercial providers: Amazon, Microsoft Azure, Google Compute Engine etc.

CLOUD COMPUTING

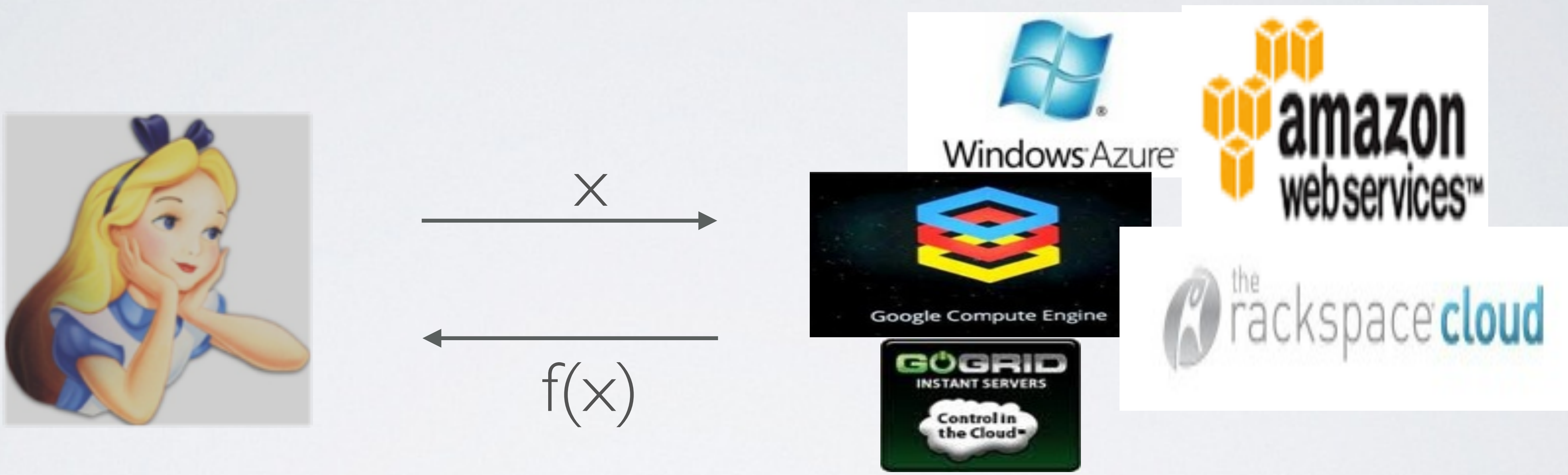
- Commercial providers: Amazon, Microsoft Azure, Google Compute Engine etc.

- Folding@Home: Stanford Project that uses computing resources of thousands of volunteer PCs/game consoles

OUTSOURCING COMPUTATION

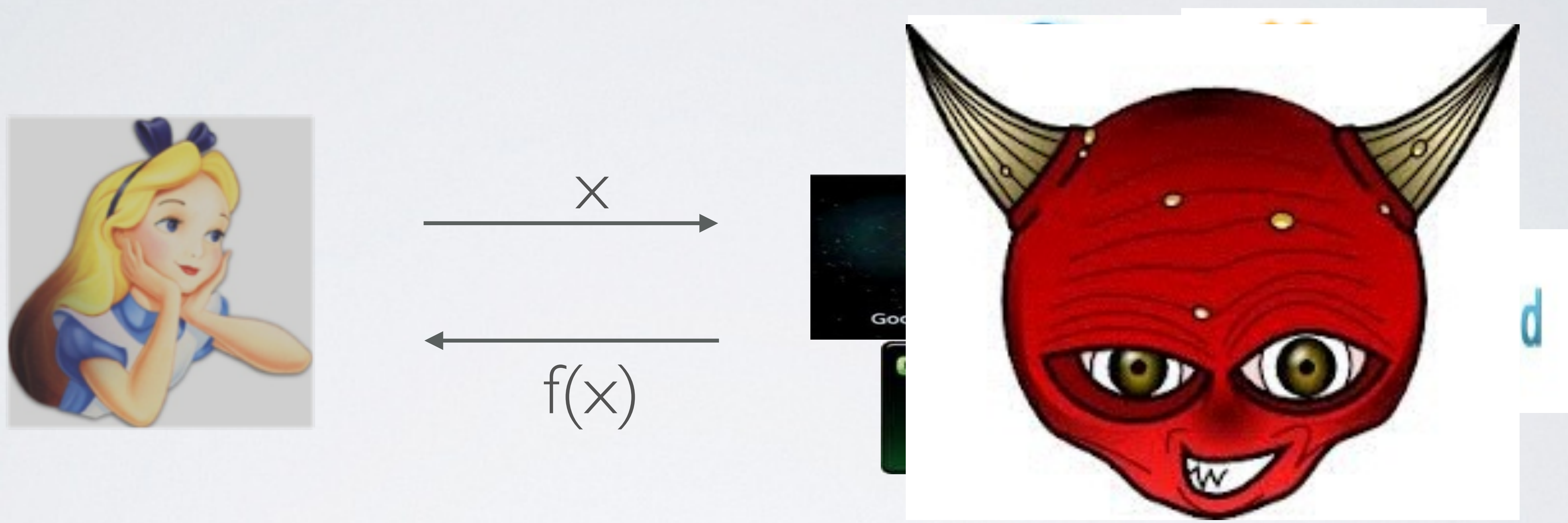


OUTSOURCING COMPUTATION



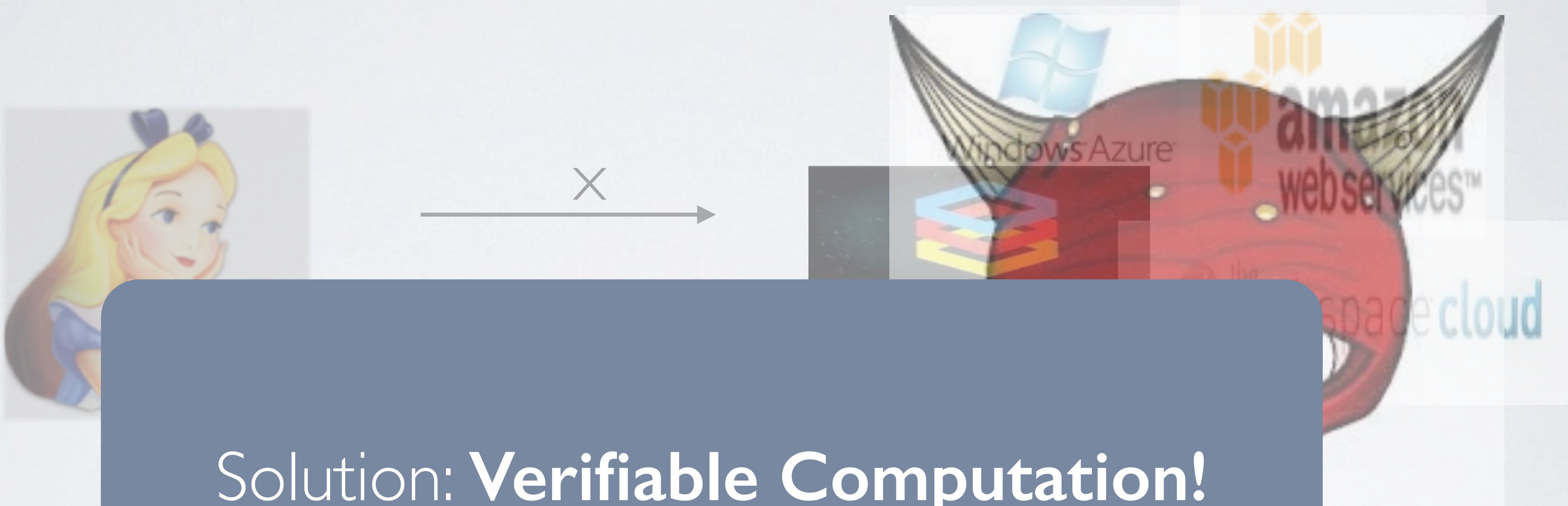
- Problem: Need to trust the companies that the computation was done correctly.

OUTSOURCING COMPUTATION



What if the cloud is malicious?

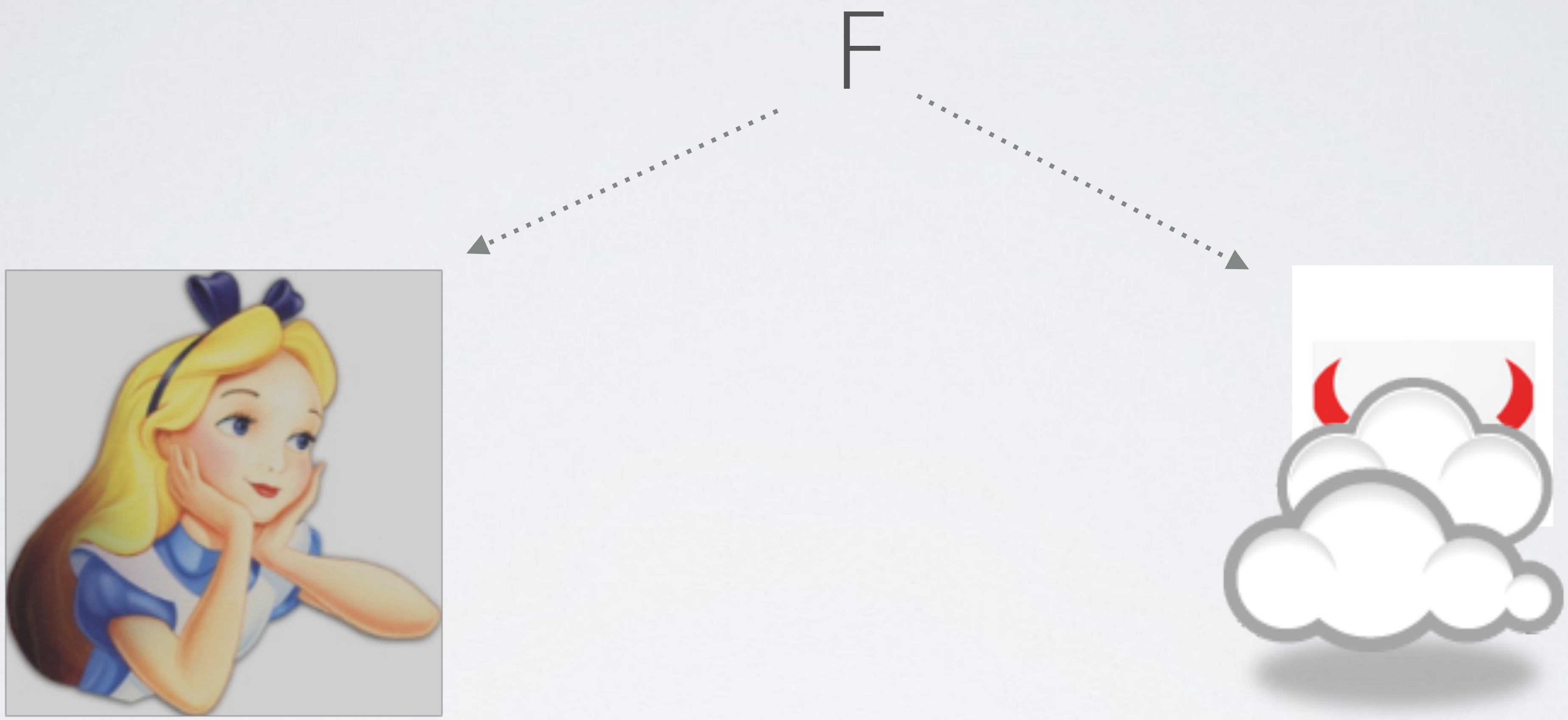
OUTSOURCING COMPUTATION



Solution: **Verifiable Computation!**

What if the cloud is malicious?

VERIFIABLE COMPUTATION



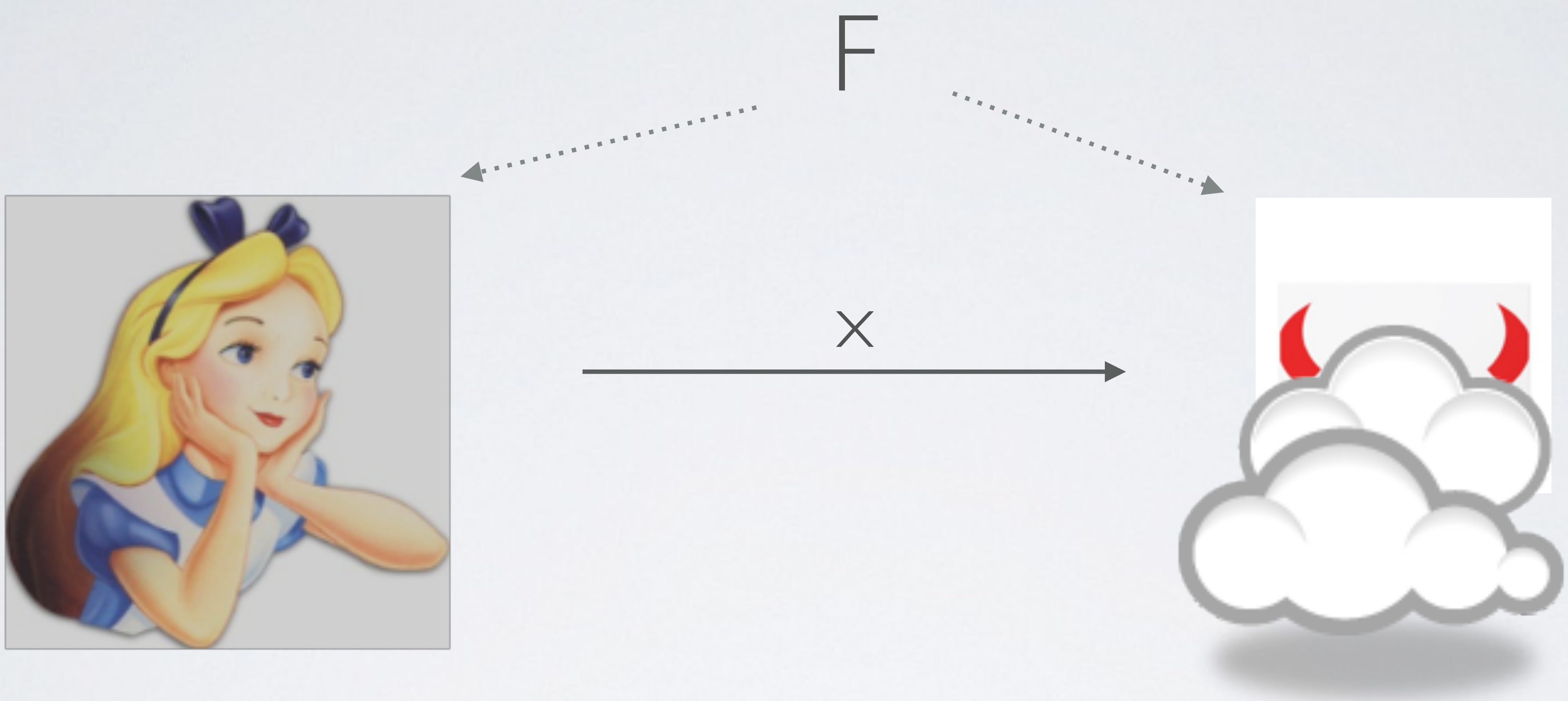
VERIFIABLE COMPUTATION

F

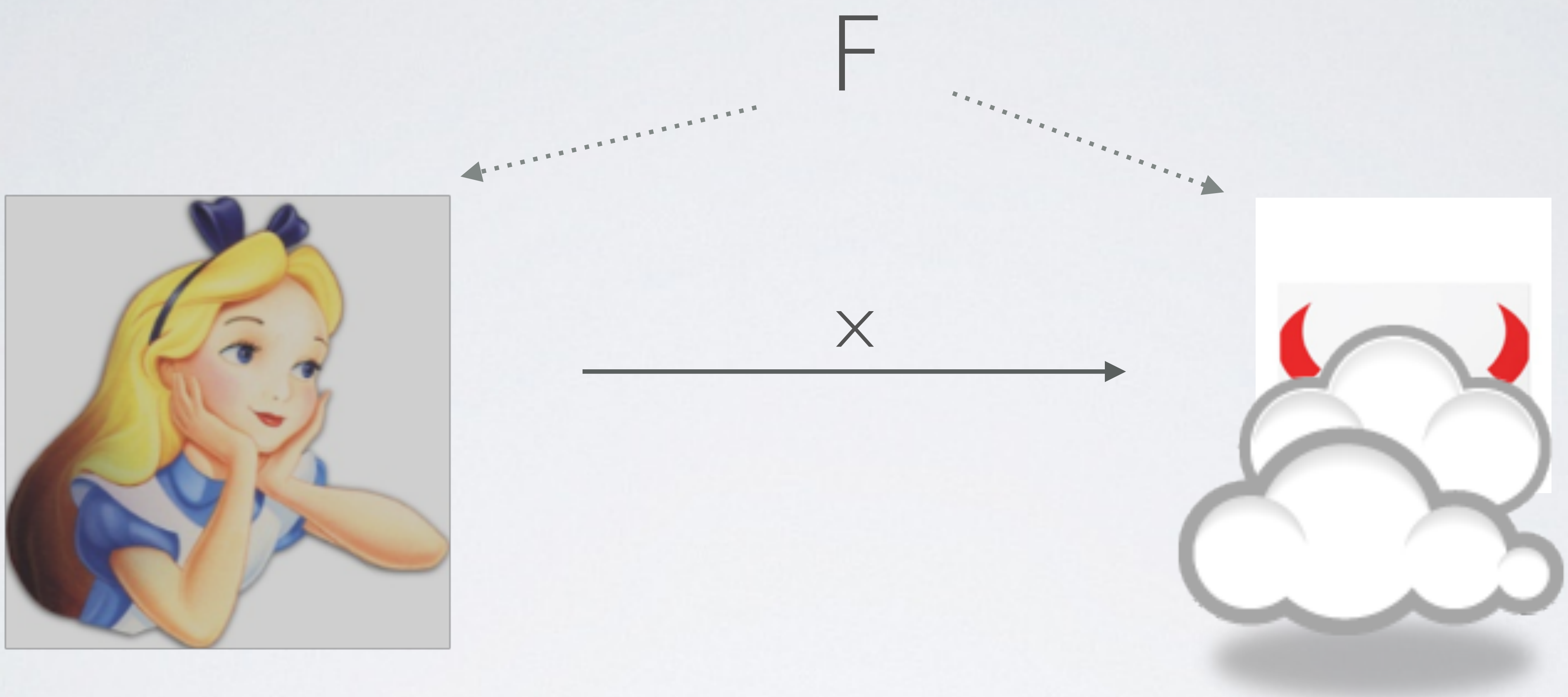
Input: x

Goal: Verifiably outsource computation of F on x to the cloud

VERIFIABLE COMPUTATION

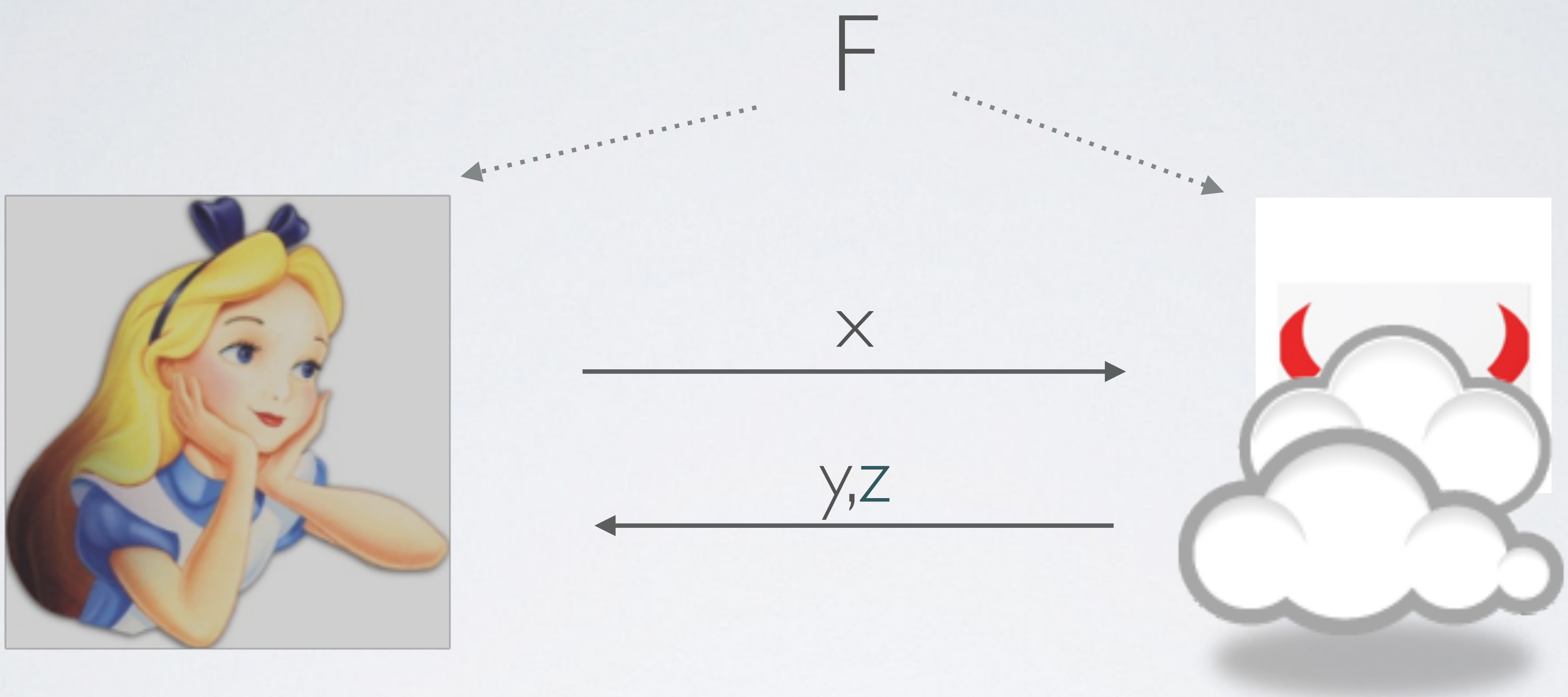


VERIFIABLE COMPUTATION



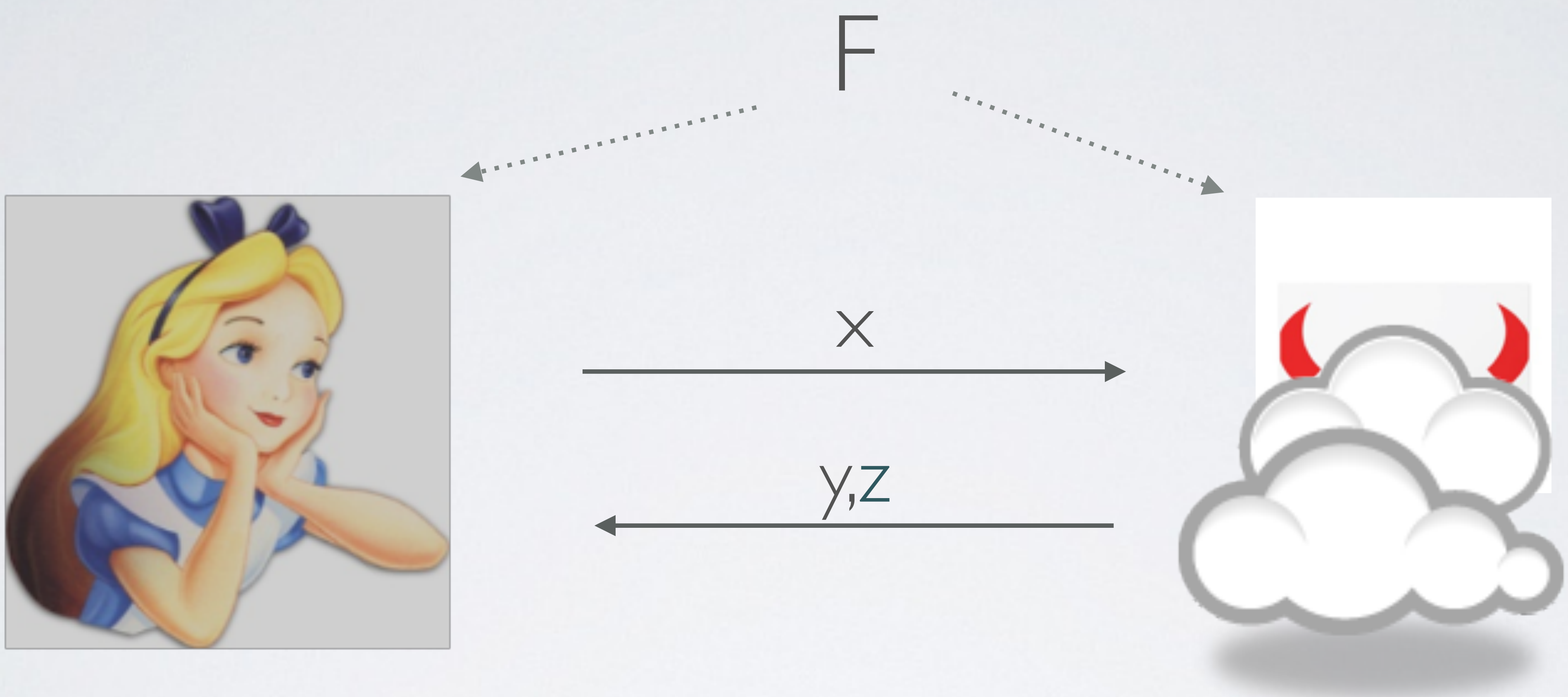
- Compute y=F(x)
- Generate proof z that the computation was done correctly

VERIFIABLE COMPUTATION



- Compute y=F(x)
- Generate proof z that the computation was done correctly

VERIFIABLE COMPUTATION



- Verify using proof z that y = F(x)

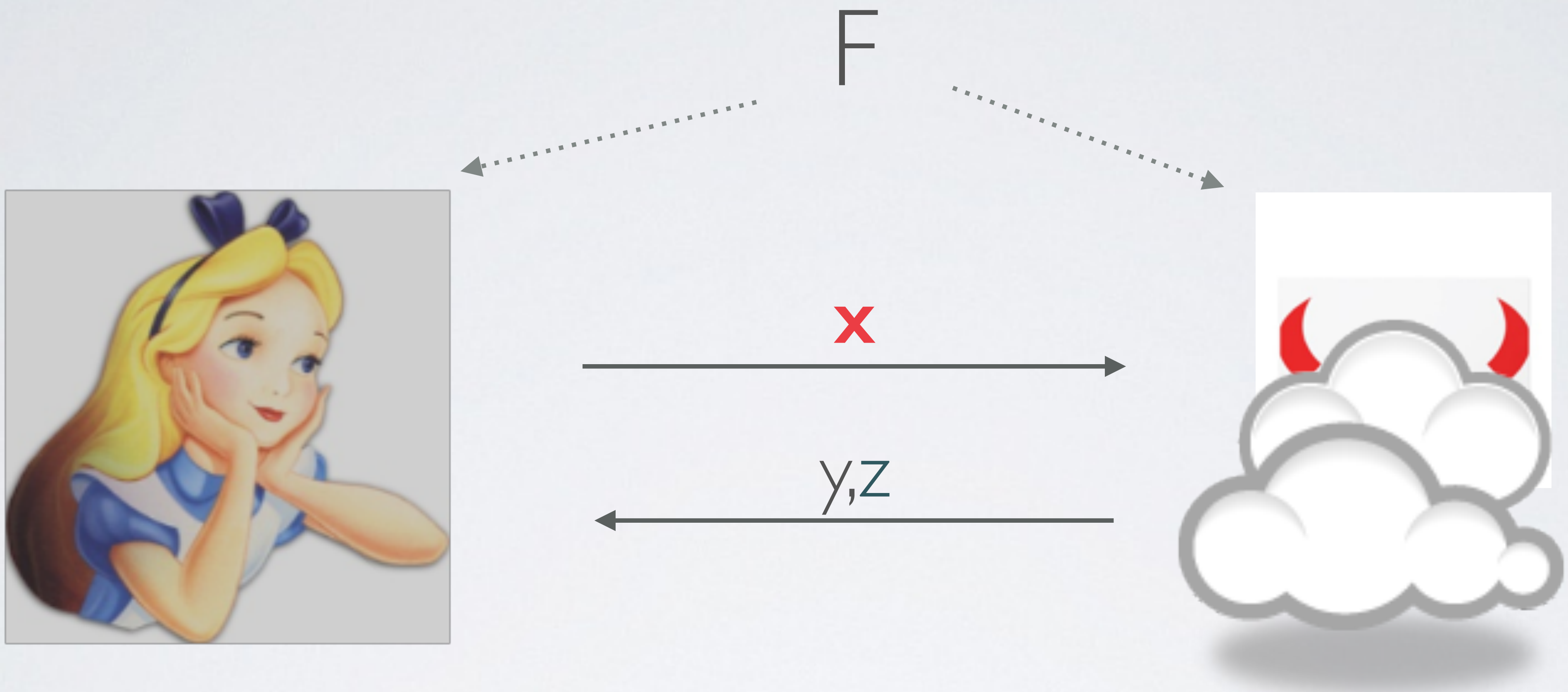
- Compute y=F(x)
- Generate proof z that the computation was done correctly

PROPERTIES OF VC PROTOCOL

- Verifiability: An adversarial cloud cannot make Alice accept an incorrect F(x).

- Efficiency: Verifying y=F(x) should be significantly easier than computing F(x) itself.

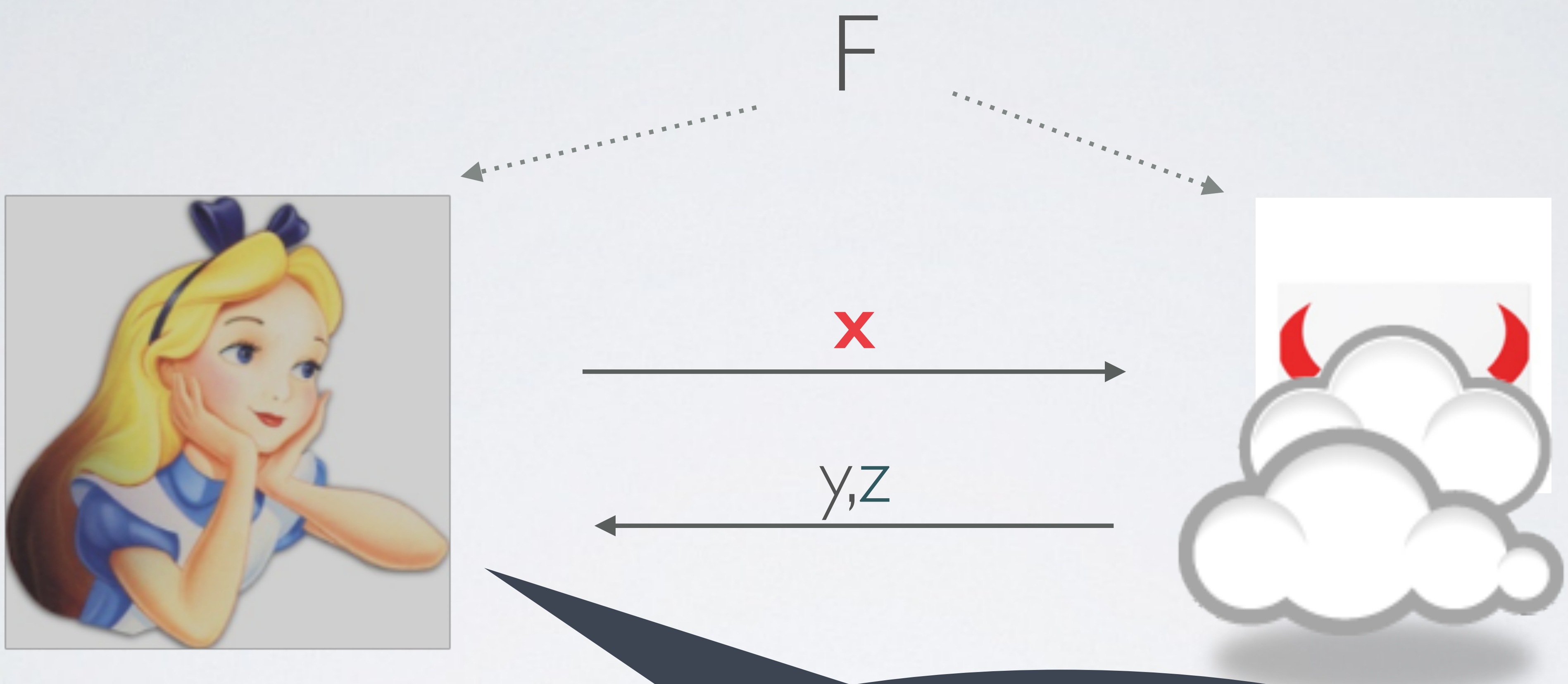
A SECOND LOOK AT VERIFIABLE COMPUTATION



- Verify using proof z that y = F(x)

- Compute y=F(x)
- Generate proof z that the computation was done correctly

A SECOND LOOK AT VERIFIABLE COMPUTATION



- Verify using proof
z that y = F(x)

Is it necessary that the cloud learns x?

F(x)
of z that
putation was
done correctly

A SECOND LOOK AT VERIFIABLE COMPUTATION

F

This work: Focus on achieving privacy

Is it necessary that the cloud learns x?

- Verify using proof z that y = F(x)

- Compute y=F(x)
- Generate proof z that the computation was done correctly

PRIOR WORK ON VC PROTOCOLS

PRIOR WORK ON VC PROTOCOLS

- No input-privacy:

- With input-privacy:

PRIOR WORK ON VC PROTOCOLS

- No input-privacy: CS proofs [Micali94], Extractable CRHF-based solutions [GLR11,BCCT11], ABE based solutions [PRV12], Quadratic Span programs [GGPR12], Information-theoretic protocols [CRR12]

- With input-privacy:

PRIOR WORK ON VC PROTOCOLS

- No input-privacy: CS proofs [Micali94], Extractable CRHF-based solutions [GLR11,BCCT11], ABE based solutions [PRV12], Quadratic Span programs [GGPR12], Information-theoretic protocols [CRR12]

- With input-privacy: FHE-based solutions [GGP10,CKV10], Randomized-Encodings based solutions [AIK10]

PRIOR WORK ON VC PROTOCOLS

- No input-privacy: CS proofs [Micali94], Extractable CRHF-based solutions [GLR11,BCCT11], ABE based solutions ... n programs [GGPR1 ... protocols [CRR12]

Either use FHE or defined for specific functions.

- With input-privacy: FHE-based solutions [GGP10,CKV10], Randomized-Encodings based solutions [AIK10]

DRAWBACKS WITH FHE

- Computational assumption: constructions under standard assumptions known only for leveled-FHE

DRAWBACKS WITH FHE

- Computational assumption: constructions under standard assumptions known only for leveled-FHE
- Efficiency: the computational overhead involved during evaluation is large.

DRAWBACKS WITH FHE

Can we achieve privacy in verifiable computation for all efficient functions without FHE?

- HE
- during evaluation is large.

ACHIEVING PRIVACY IN VC WITHOUT FHE

- Single server case: FHE seems to be inherently required.

ACHIEVING PRIVACY IN VC WITHOUT FHE

- Single server case: FHE seems to be inherently required.

We focus on the case when Alice outsources her computation to many servers

RESULTS

RESULTS

- 2-server case: based on one-way functions, very efficient

RESULTS

- 2-server case: based on one-way functions, very efficient

- n-servers:

  Protocol #1: can tolerate (n-1) dishonest servers, based on DDH assumption

RESULTS

- 2-server case: based on one-way functions, client is very efficient

- n-servers:

  Protocol #1: can tolerate (n-1) dishonest servers, based on DDH assumption

  Protocol #2: can tolerate constant fraction of dishonest servers, based on one-way functions

RESULTS

- 2-server case: based on one-way functions, client is very efficient

- n-servers:

  Protocol #1: can tolerate (n-1) dishonest servers, based on DDH assumption

  Protocol #2: can tolerate constant fraction of dishonest servers, based on one-way functions

PROPERTIES OF PROTOCOL #1

- Based on DDH assumption

PROPERTIES OF PROTOCOL #1

- Based on DDH assumption
- At least one honest server

PROPERTIES OF PROTOCOL #1

- Based on DDH assumption
- At least one honest server
- No preprocessing

PROPERTIES OF PROTOCOL #1

- Based on DDH assumption
- At lea
- No preprocessing

Many prior known protocols : expensive preprocessing

PROPERTIES OF PROTOCOL #1

- Based on DDH assumption
- At least one honest server
- No preprocessing
- The client complexity independent of the function complexity.

PROPERTIES OF PROTOCOL #1

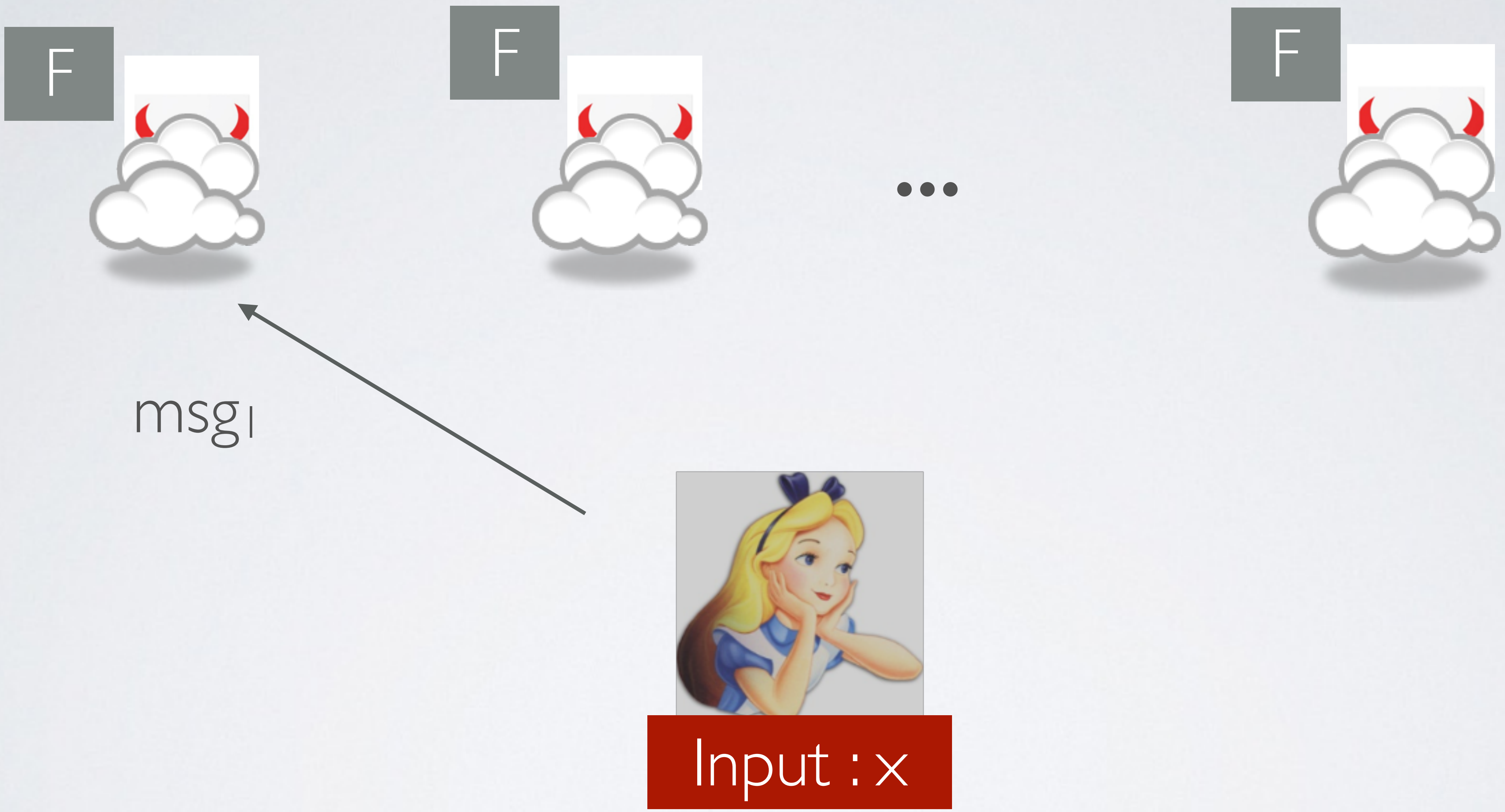
- At least one honest server
- No pre
- The client complexity independent of the function complexity.

In all prior known protocols : the client complexity depends on the function complexity

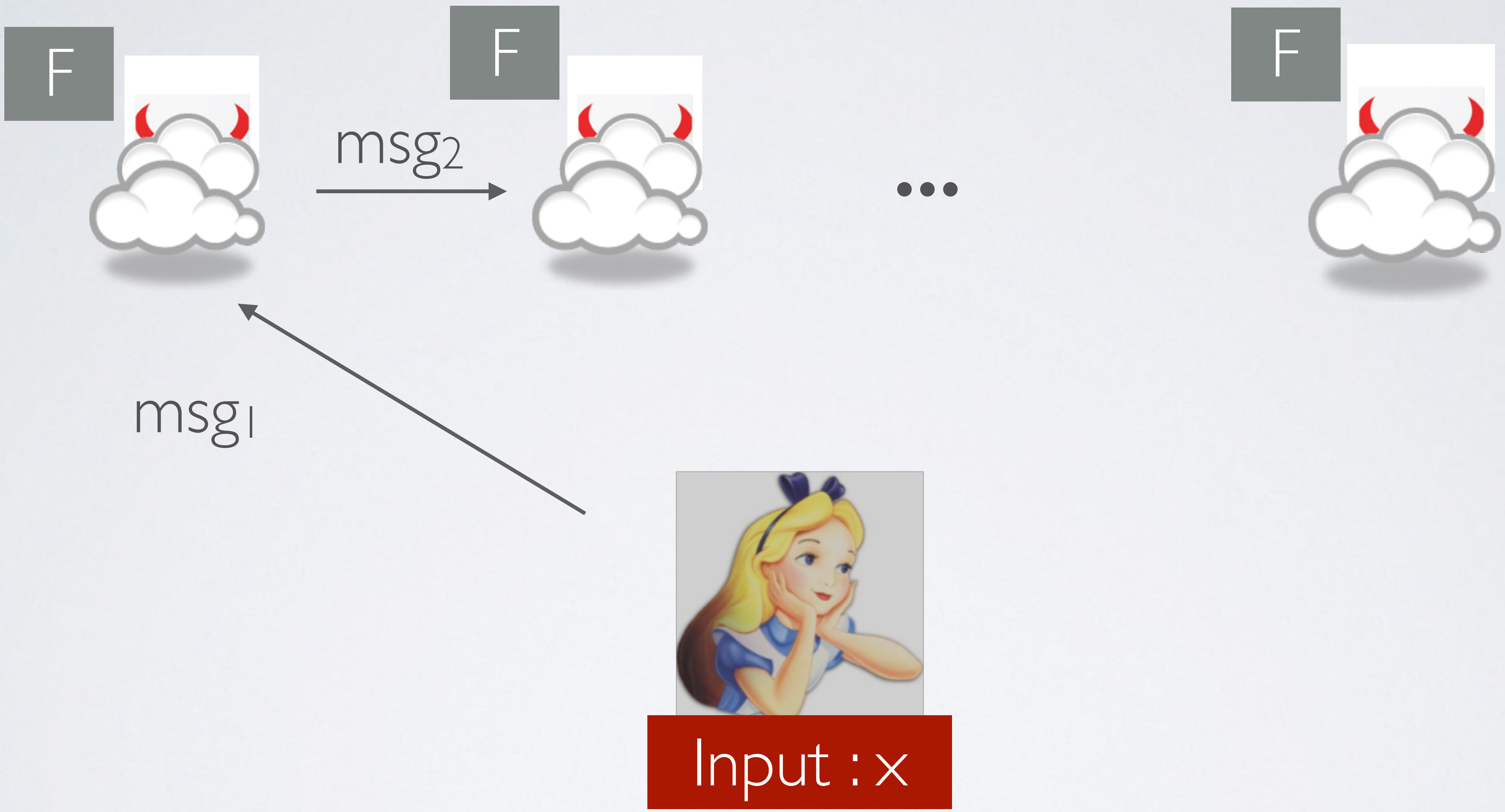
TECHNICAL DETAILS

VC IN THE MULTIPLE-SERVER MODEL

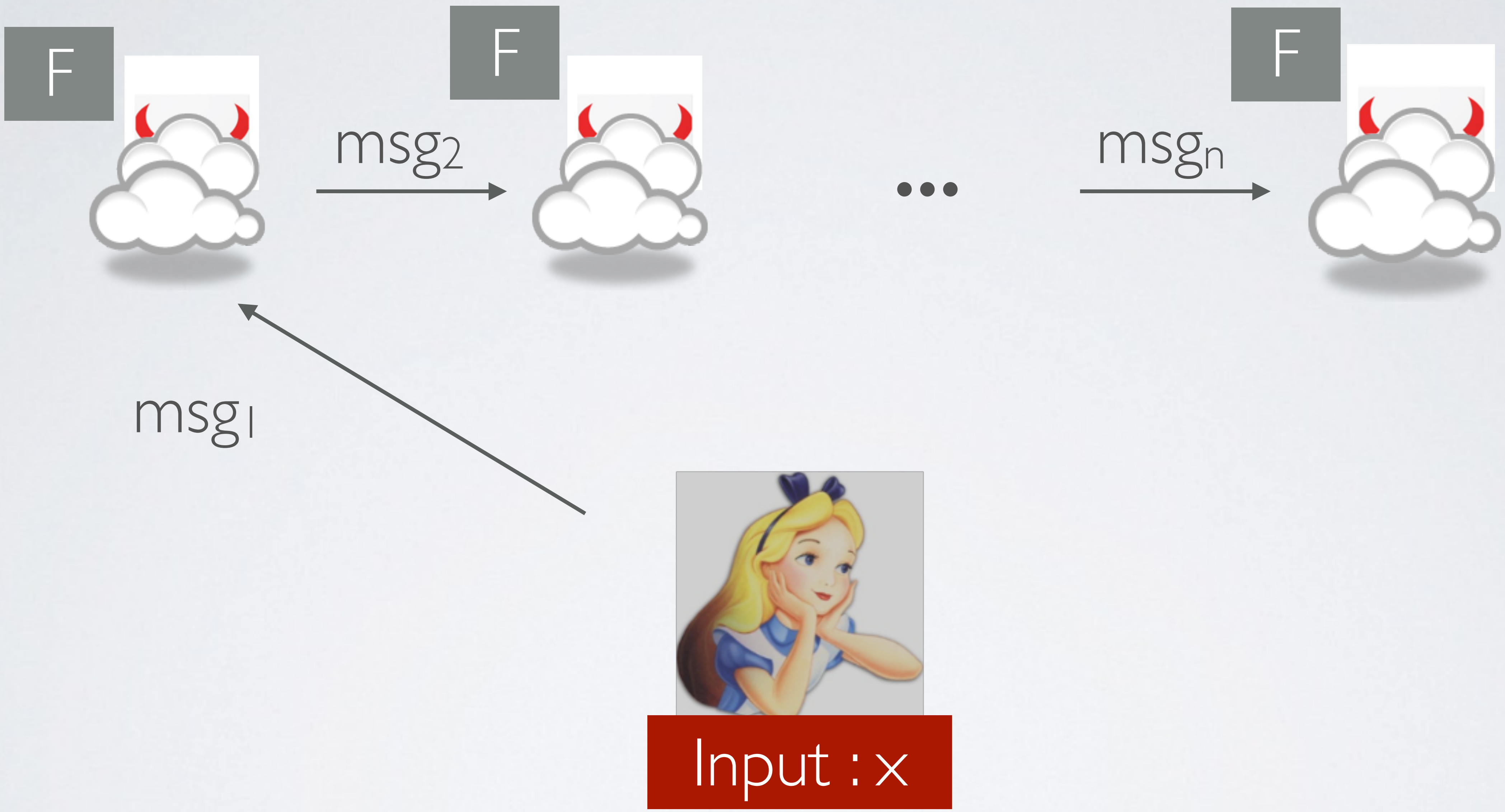
VC IN THE MULTIPLE-SERVER MODEL



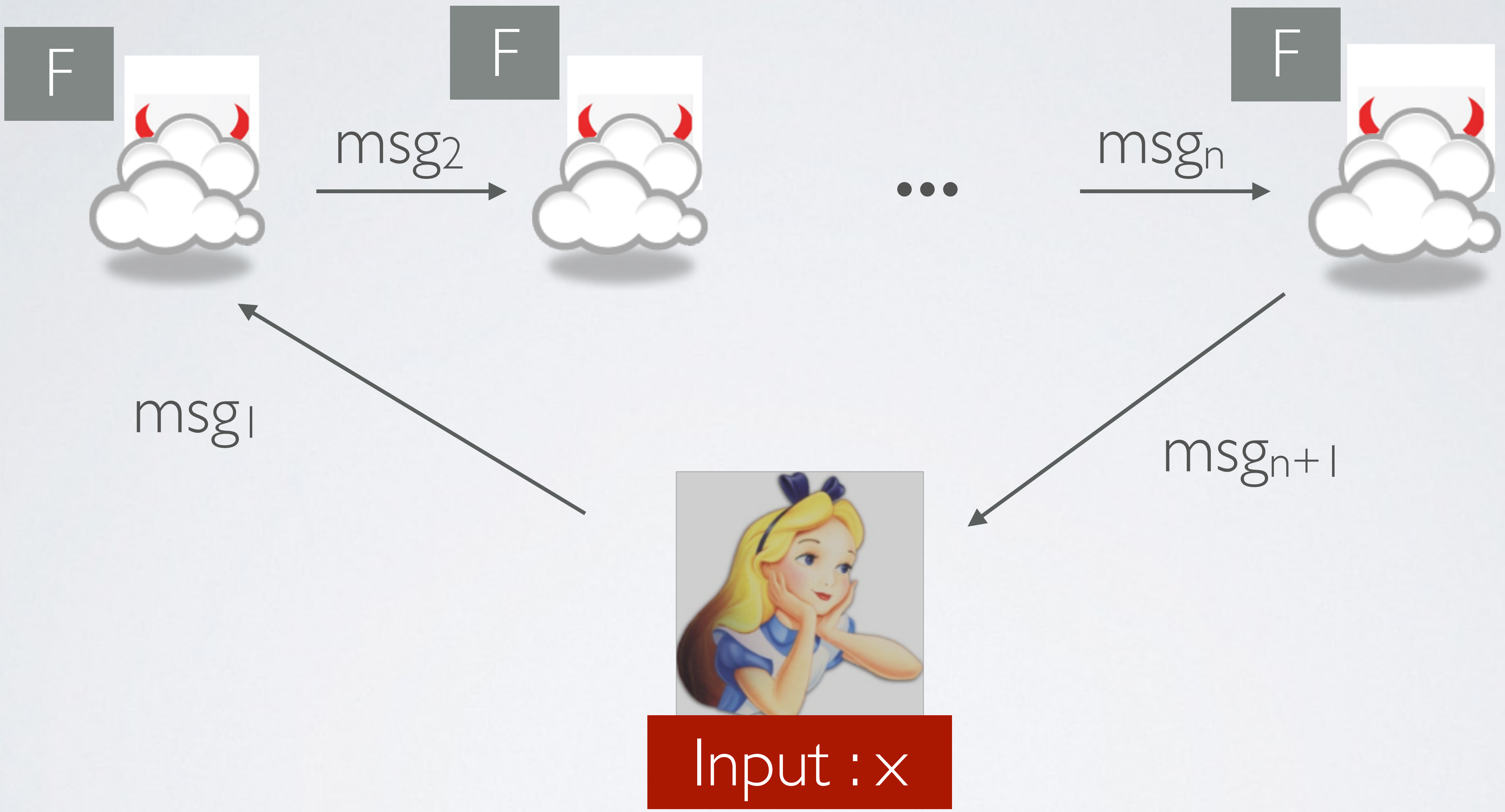
VC IN THE MULTIPLE-SERVER MODEL



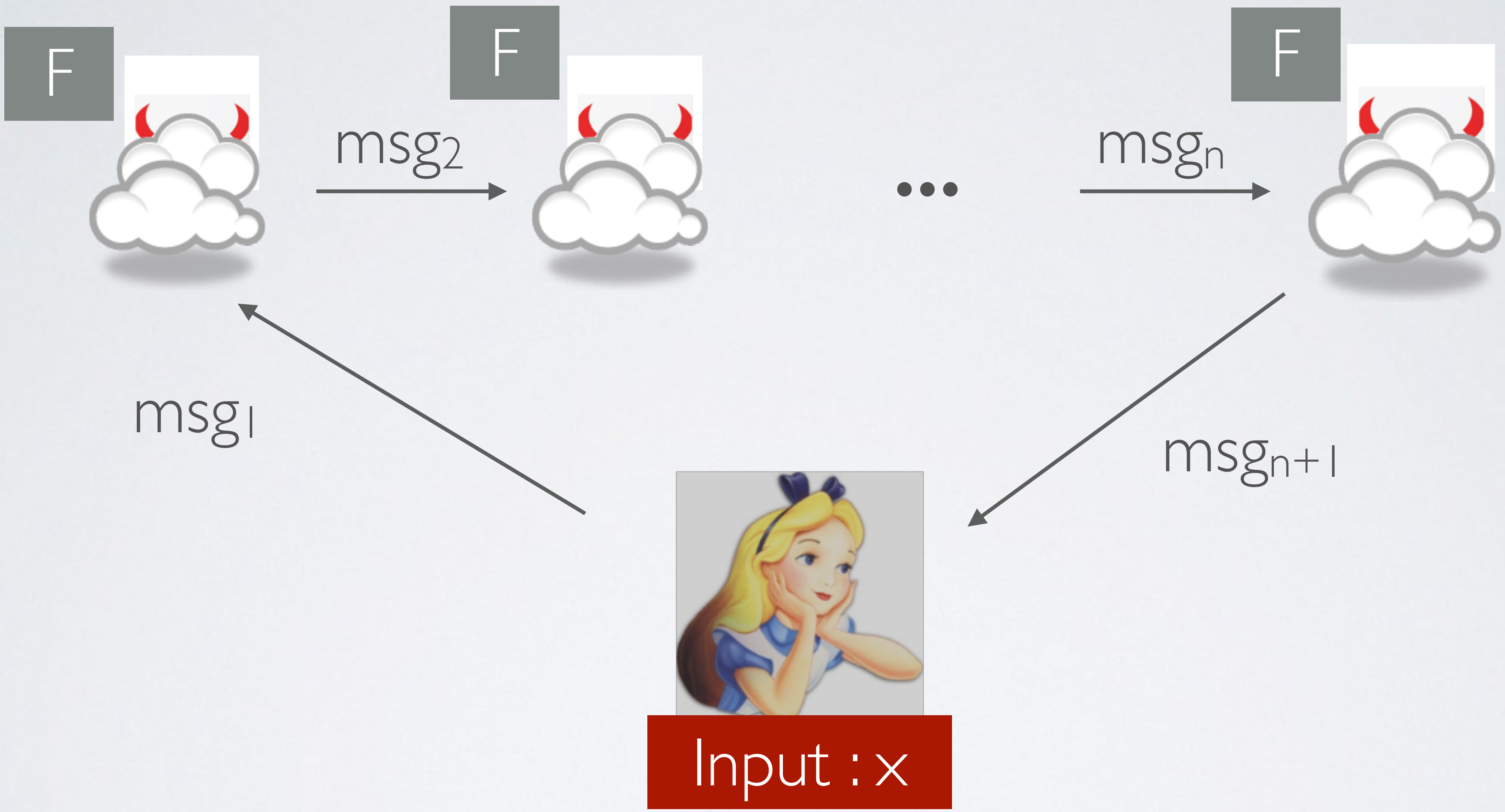
VC IN THE MULTIPLE-SERVER MODEL



VC IN THE MULTIPLE-SERVER MODEL

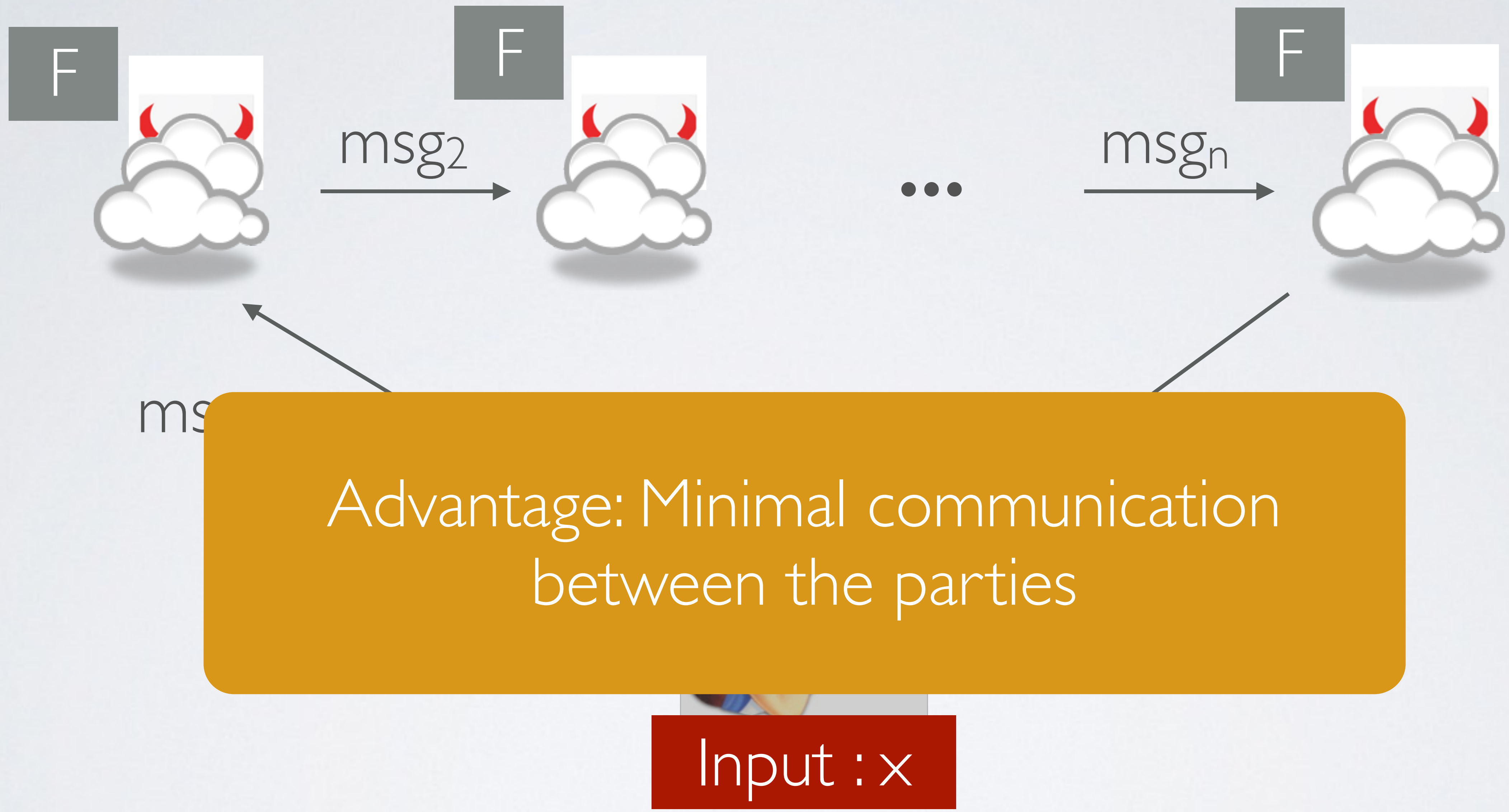


VC IN THE MULTIPLE-SERVER MODEL



- Alice retrieves F(x) from msg_{n+1}

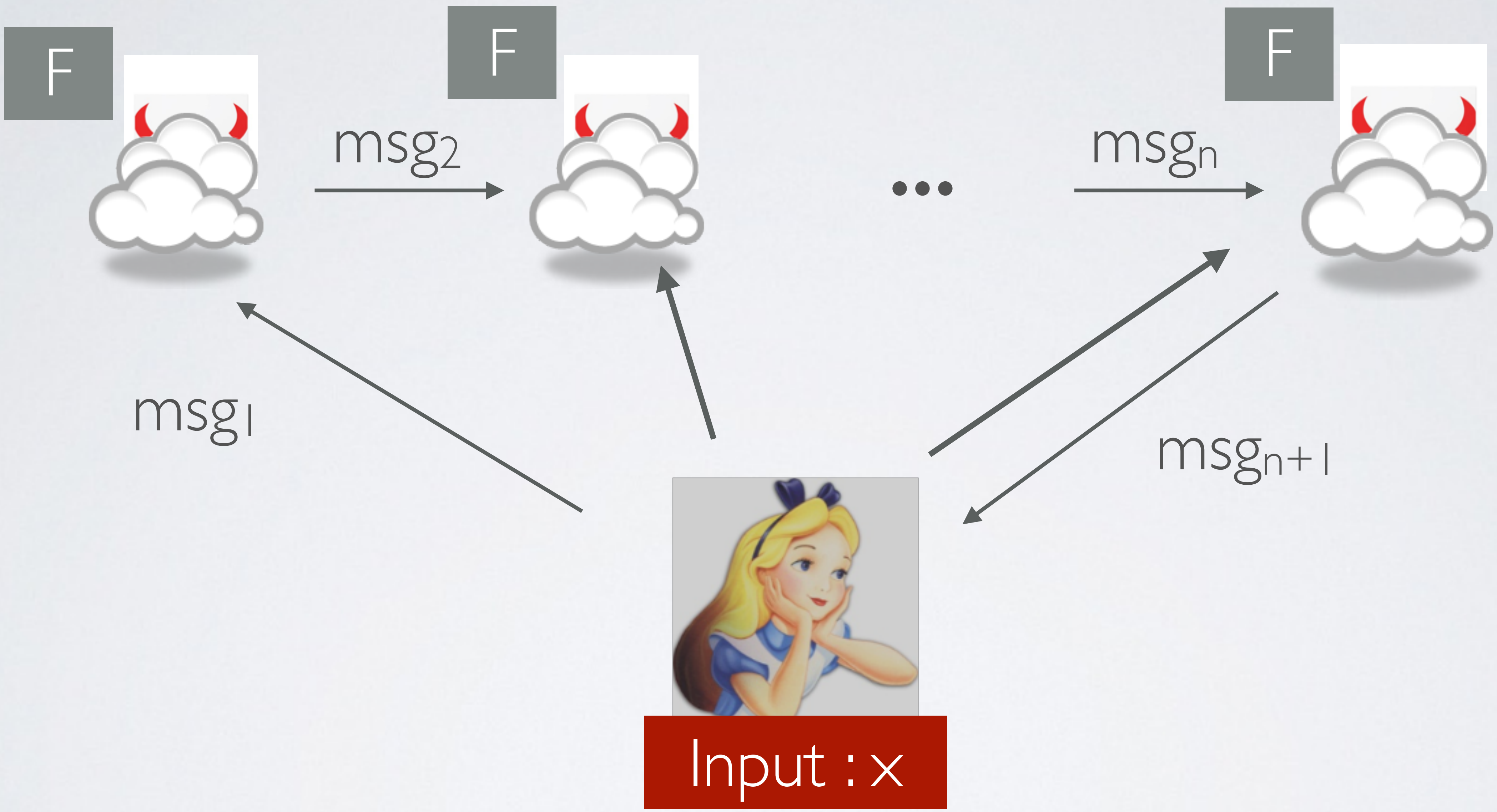
VC IN THE MULTIPLE-SERVER MODEL



Advantage: Minimal communication between the parties

- Alice retrieves F(x) from msg_{n+1}

THIS TALK: EQUIVALENT MODEL



- Alice sends messages to intermediate servers

THIS TALK: EQUIVALENT MODEL

This is equivalent to the previous model:

- Client encrypts all the messages with public keys of servers
- Signs the ciphertexts
- Sends the ciphertexts to the first server.

- Alice sends messages to intermediate servers

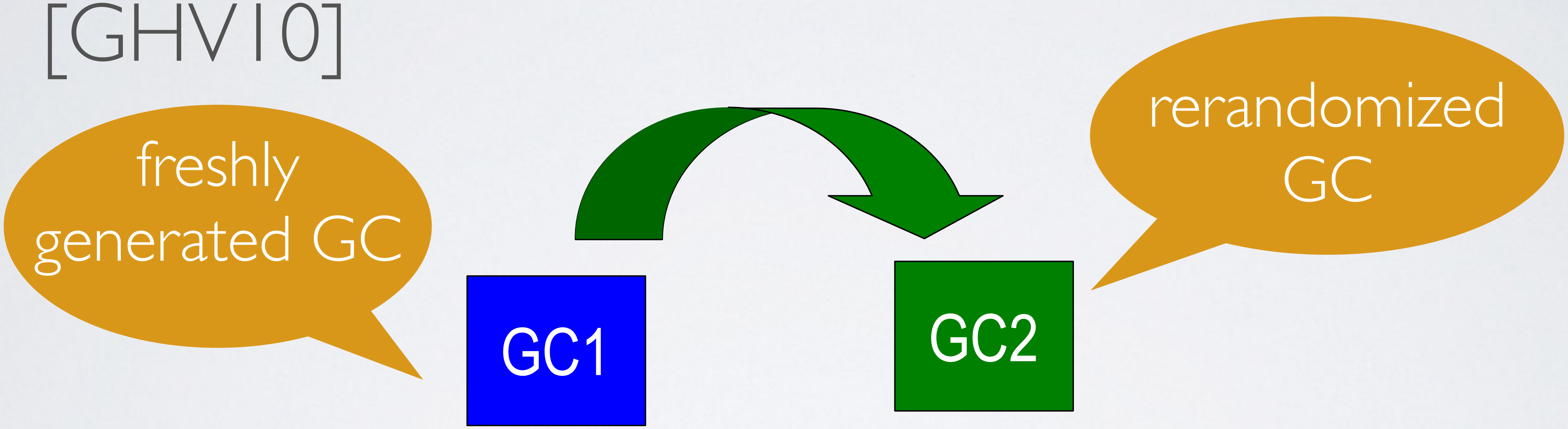
OUR CONSTRUCTION

OUR CONSTRUCTION

- Main ingredient: rerandomizable garbled circuits [GHV10]

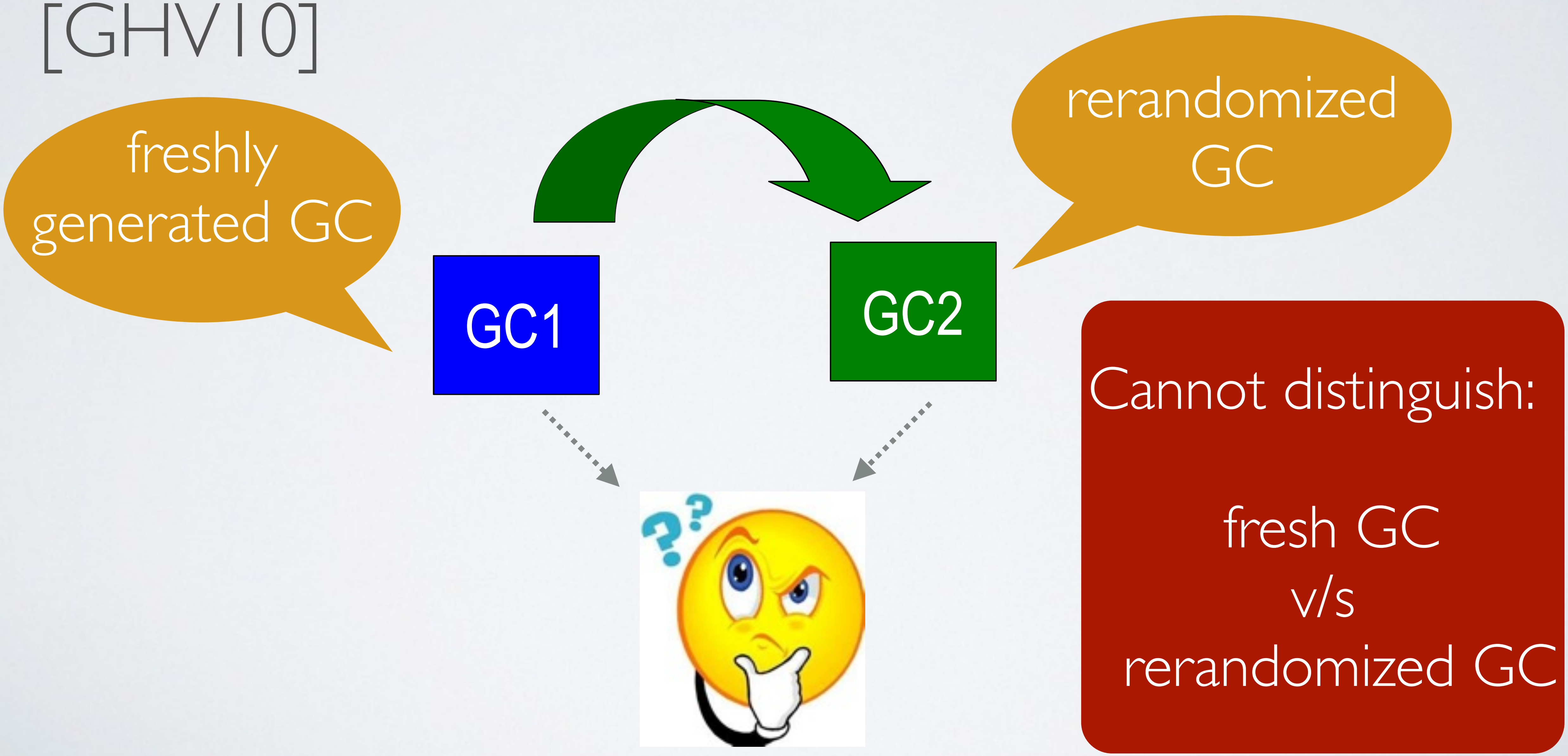
OUR CONSTRUCTION

- Main ingredient: rerandomizable garbled circuits [GHV10]



OUR CONSTRUCTION

- Main ingredient: rerandomizable garbled circuits [GHV10]



OUR CONSTRUCTION

- Main ingredient: rerandomizable garbled circuits [GHV10]

- Use encryption scheme that has homomorphic properties.

OUR CONSTRUCTION

- Main ingredient: rerandomizable garbled circuits [GHV10]

  - Use encryption scheme that has homomorphic properties.

Supports permutation and XOR

USING RERANDOMIZABLE GC

- First server: garbles the circuit F to obtain GC_1 and sends to second server.
- i^{th} server: rerandomizes the garbled circuit GC_{i-1} to obtain GC_i and sends to $i+1^{th}$ server.
- Last server: evaluates the garbled circuit to obtain F(x).

USING RERANDOMIZABLE GC

- First server: garbles the circuit F to obtain GC_1 and sends to second server.
- i^{th} server: re circuit GC_{i-1} to obtain GC_i er.

wire keys supplied by the client

- Last server: evaluates the garbled circuit to obtain F(x).

USING RERANDOMIZABLE GC: ISSUES

- The garbled circuits can be maliciously generated by servers

USING RERANDOMIZABLE GC: ISSUES

- The garbled circuits can be maliciously generated by servers

  - Use NIZKs

USING RERANDOMIZABLE GC: ISSUES

- The garbled circuits can be maliciously generated by servers

  - Use NIZKs

- The servers can use improper randomness

USING RERANDOMIZABLE GC: ISSUES

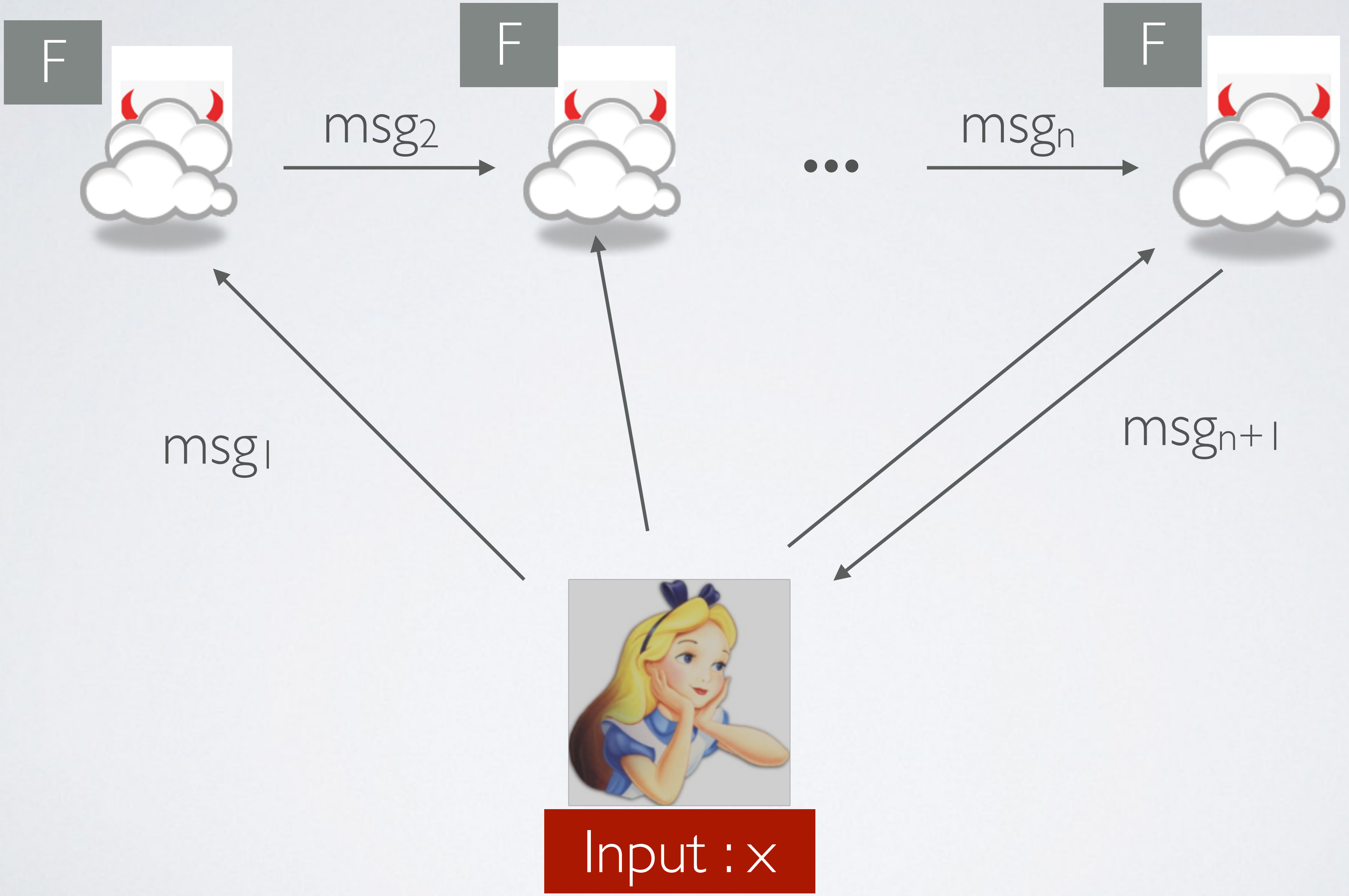
- The garbled circuits can be maliciously generated by servers

  - Use NIZKs

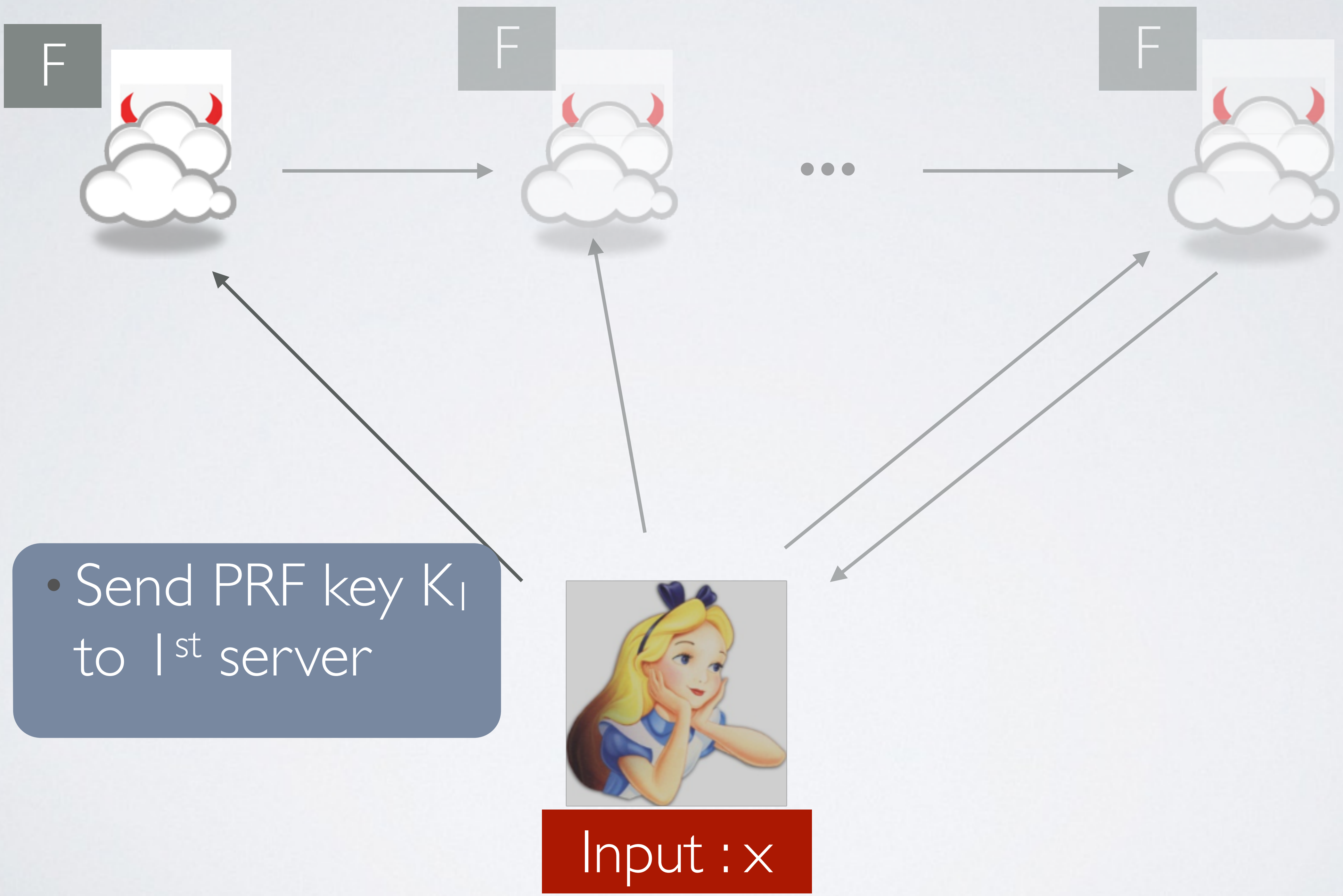
- The servers can use improper randomness

  - Client gives PRF keys to the servers

OUR CONSTRUCTION

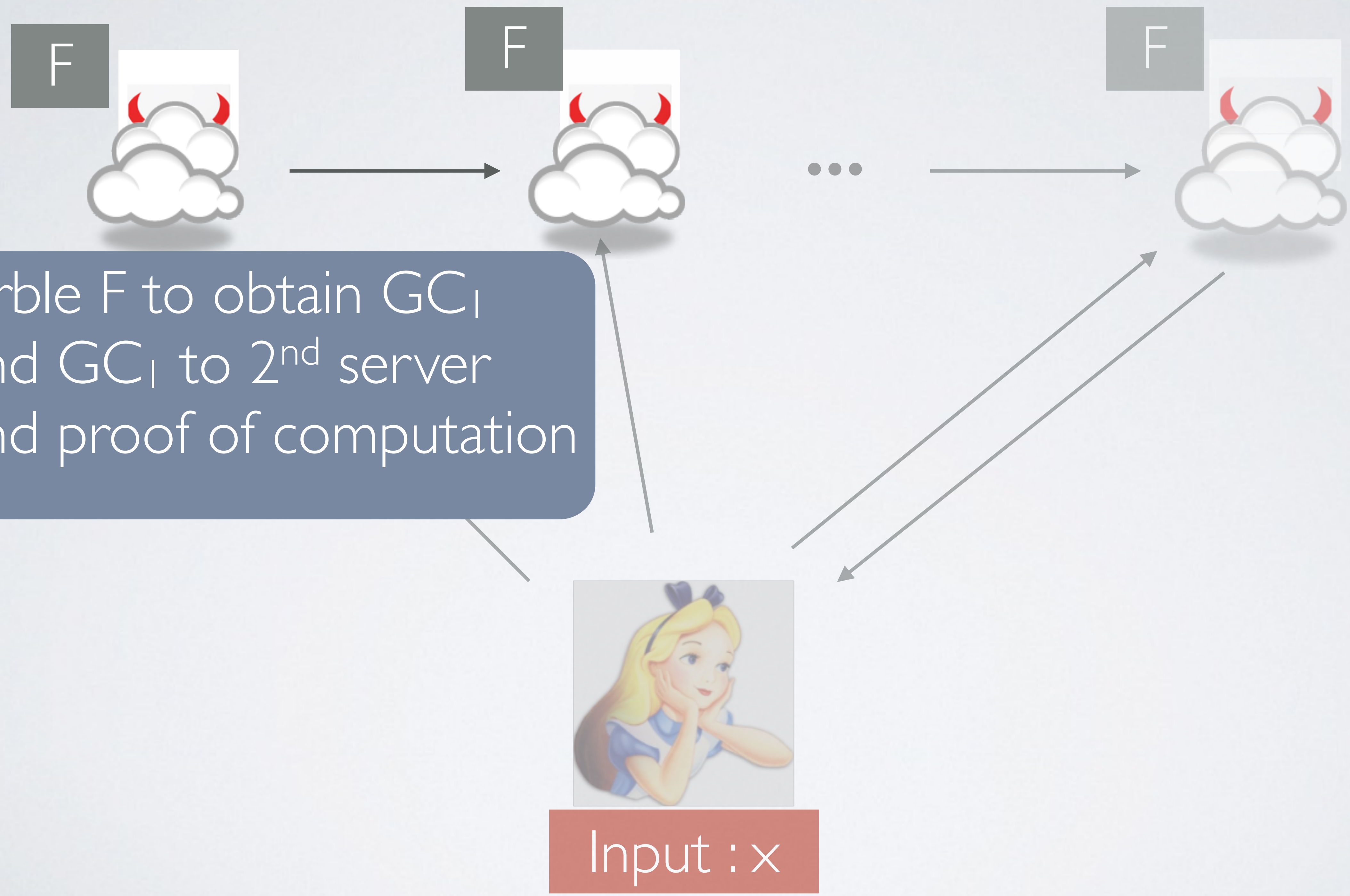


OUR CONSTRUCTION



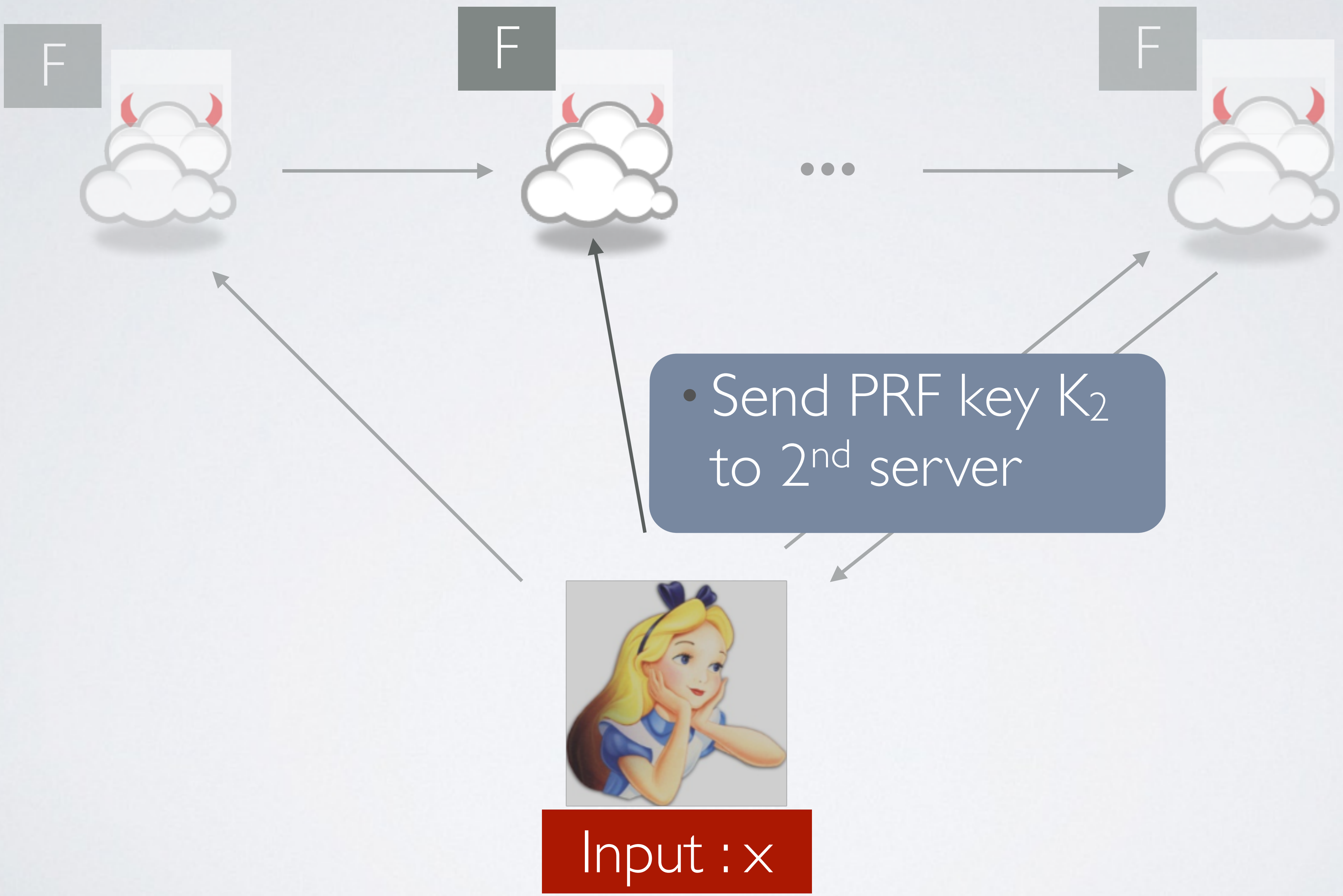
- Send PRF key K_1 to 1^{st} server

OUR CONSTRUCTION

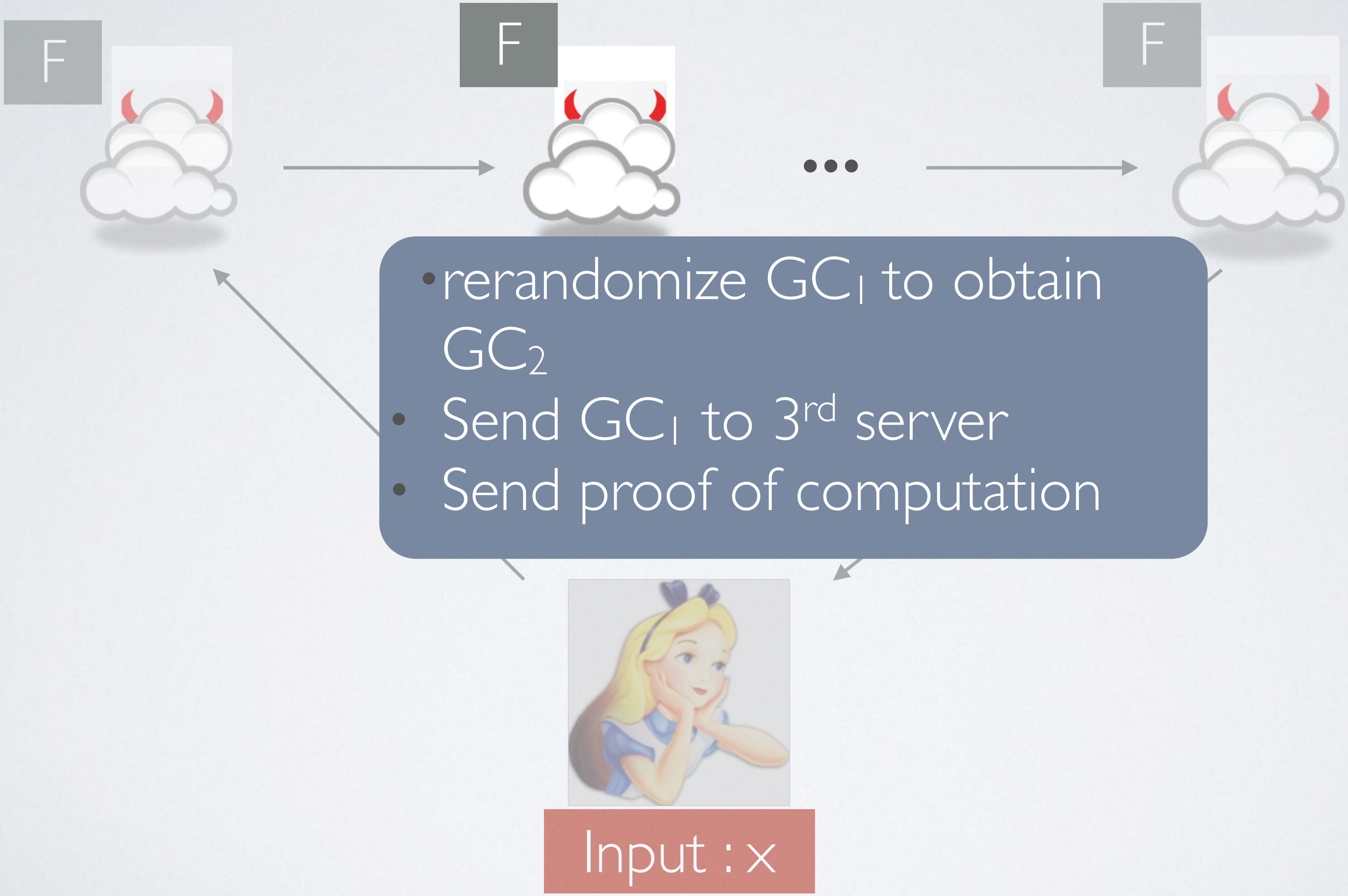


- Garble F to obtain GC_1
- Send GC_1 to 2^{nd} server
- Send proof of computation

OUR CONSTRUCTION

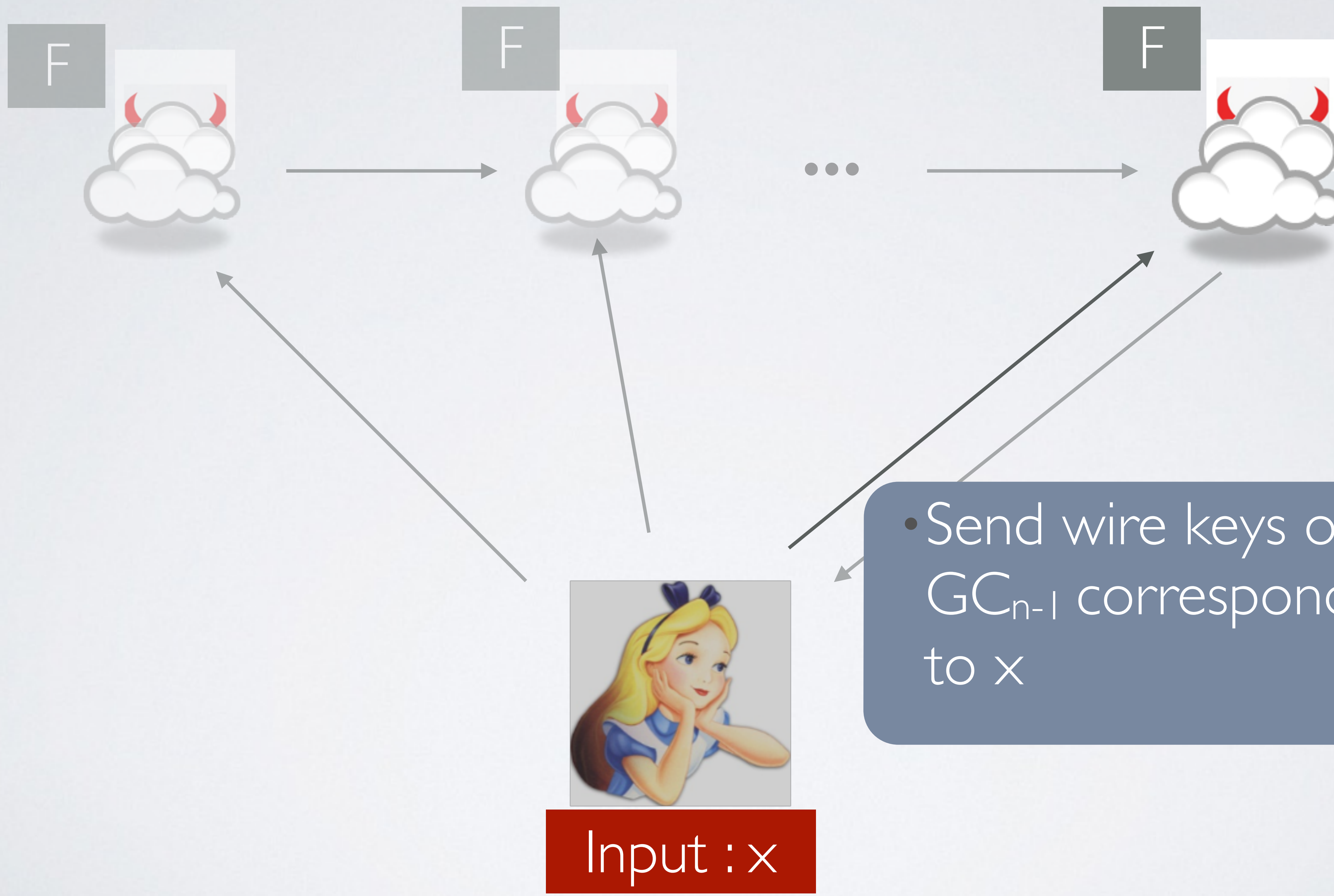


OUR CONSTRUCTION



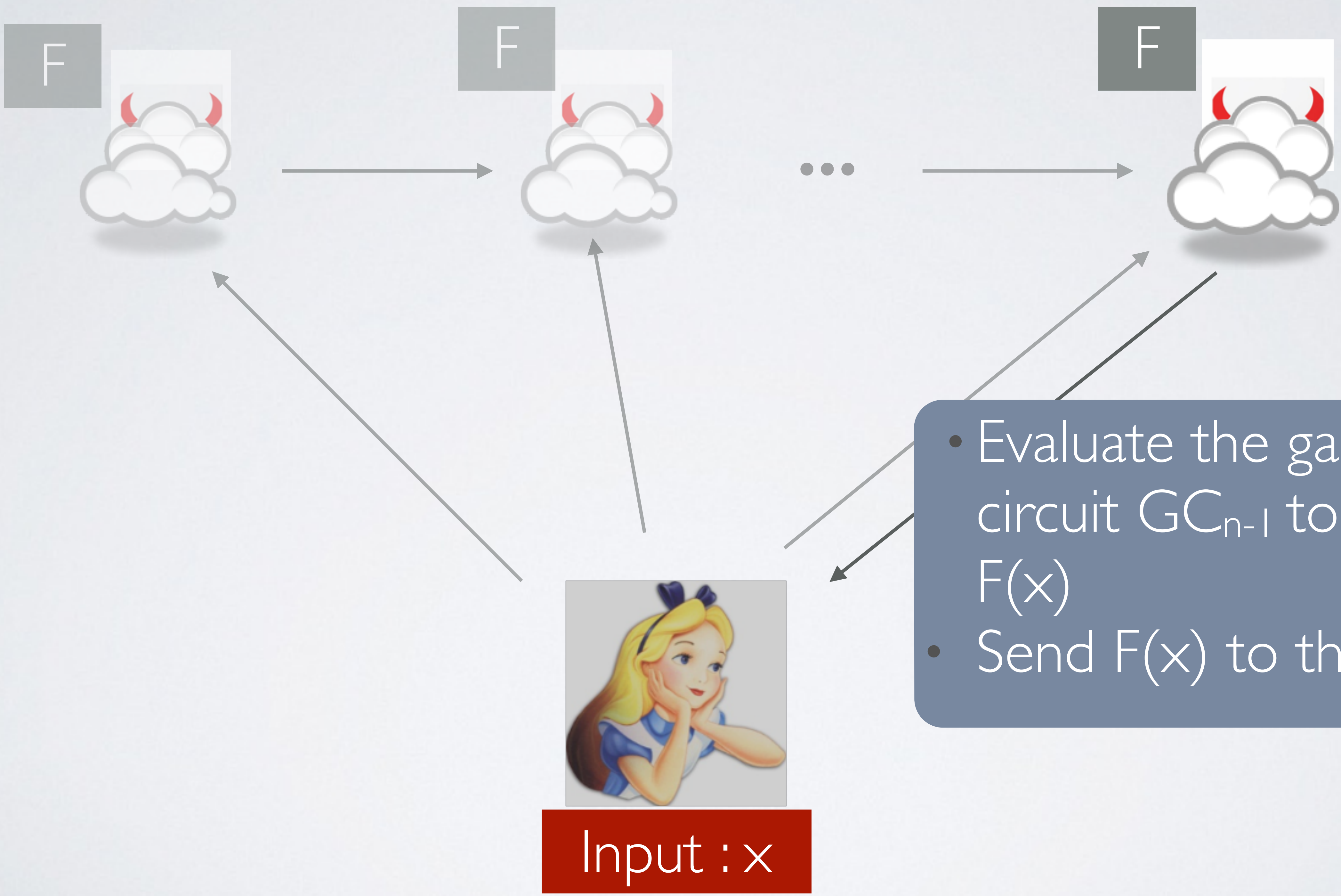
- rerandomize GC_1 to obtain GC_2
- Send GC_1 to 3rd server
- Send proof of computation

OUR CONSTRUCTION



- Send wire keys of GC_{n-1} corresponding to x

OUR CONSTRUCTION



- Evaluate the garbled circuit GC_{n-1} to obtain F(x)
- Send F(x) to the client

OUR CONSTRUCTION: PROPERTIES

OUR CONSTRUCTION: PROPERTIES

- Input-privacy? YES

OUR CONSTRUCTION: PROPERTIES

- Input-privacy? YES
- Client efficiency? YES

OUR CONSTRUCTION: PROPERTIES

- Input-privacy? YES
- Client efficiency? YES
- Verifiability? NO!

OUR CONSTRUCTION: PROPERTIES

- Input-privacy? YES
- Client efficiency? YES
- Verifiability? NO!

The last server might send an incorrect value as F(x)

ENSURING VERIFIABILITY

- Consider the function $G(\cdot,\cdot)$:

$$G(x,K) \text{ outputs } \big(F(x), MAC(K,F(x))\big)$$

ENSURING VERIFIABILITY

- Consider the function G(·,·):

$$G(x,K) \text{ outputs } \big(F(x), MAC(K,F(x))\big)$$

- Modify our construction as follows:

  - Garble G(·,·) instead of F

  - Client sends wire keys corresponding to x and K instead of just x

REMARK ON CLIENT COMPLEXITY

- Client complexity depends on the number of servers

REMARK ON CLIENT COMPLEXITY

- Client complexity depends on the number of servers

  - Has to send PRF keys to all servers.

REMARK ON CLIENT COMPLEXITY

- Client complexity depends on the number of servers

  - Has to send PRF keys to all servers.

  - Has to rerandomize the wire keys (n-2) times to obtain wire keys of the last GC

REMARK ON CLIENT COMPLEXITY

- Client complexity depends on the number of servers

  - Has to send PRF keys to all servers.

  - Has to re[...]imes to obtain w[...]

Using preprocessing: generate all PRF keys during preprocessing phase

REMARK ON CLIENT COMPLEXITY

- Client complexity depends on the number of servers

  - Has to send PRF keys to all servers.

  - Has to rerandomize the wire keys (n-2) times to obtain wire keys of the last GC

???

IMPROVING CLIENT COMPLEXITY

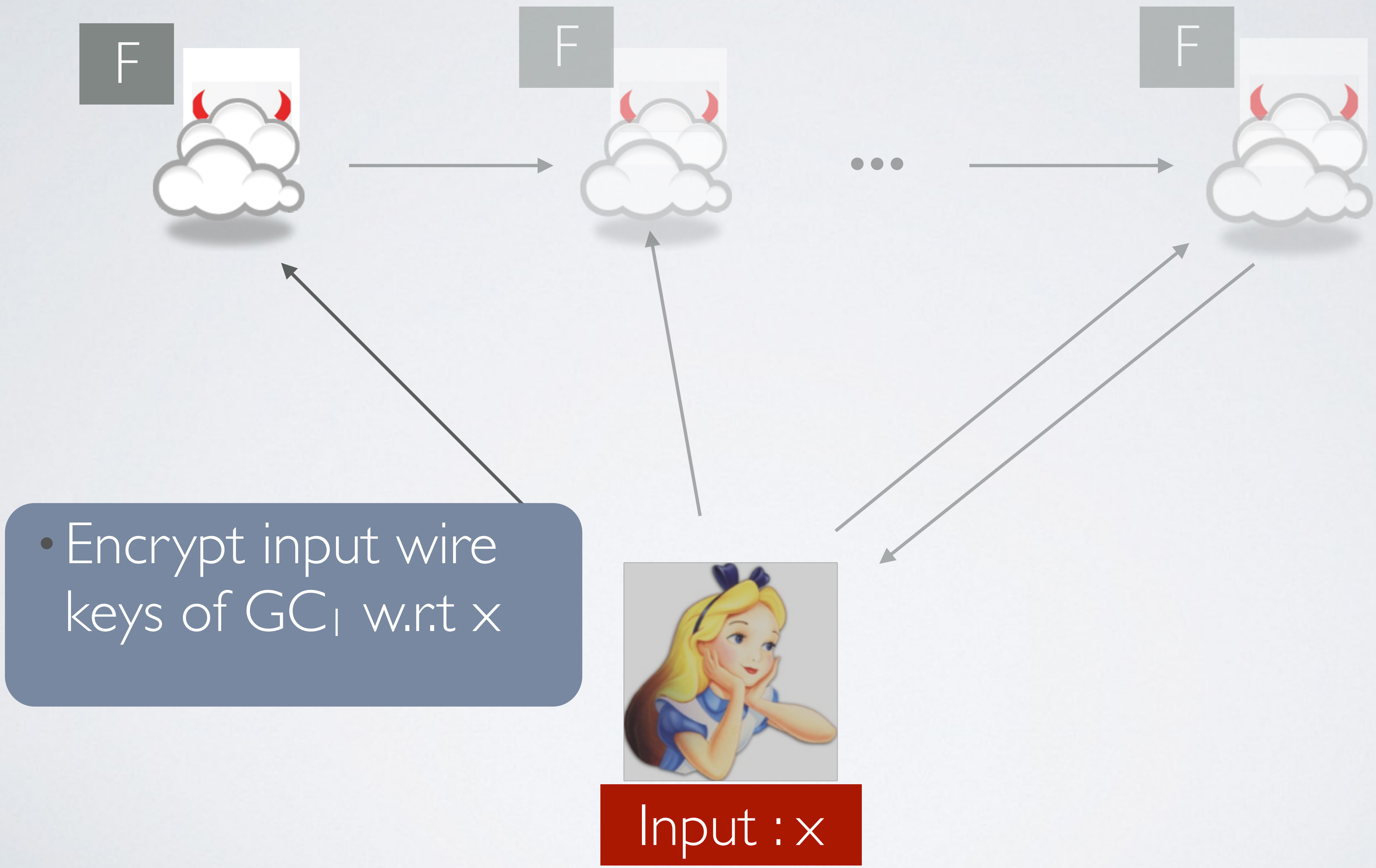
- Warmup attempt: Using encryption scheme having homomorphic properties.

IMPROVING CLIENT COMPLEXITY

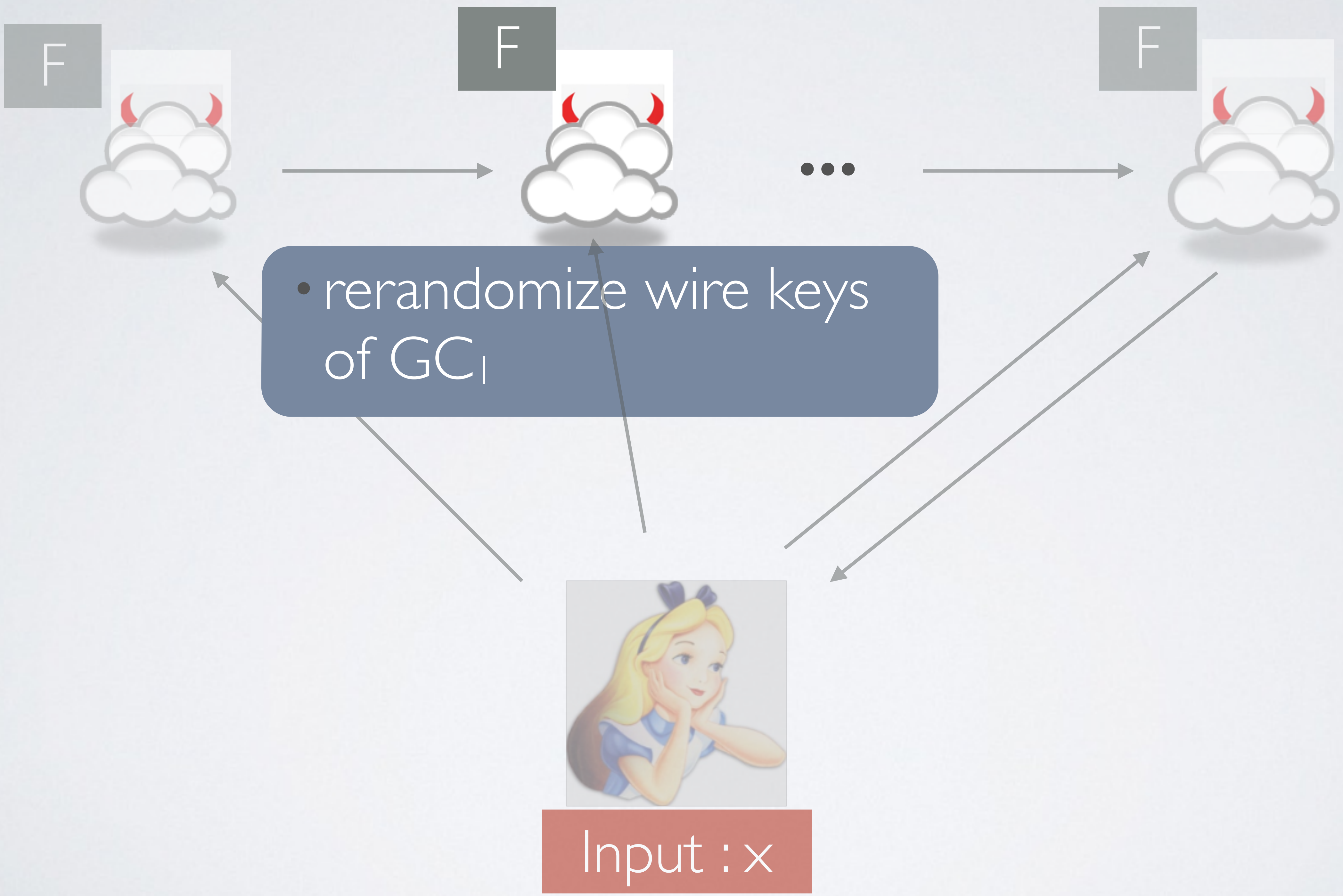
- Warmup attempt: Using encryption scheme having homomorphic properties.

Supporting permutation and XOR operations.

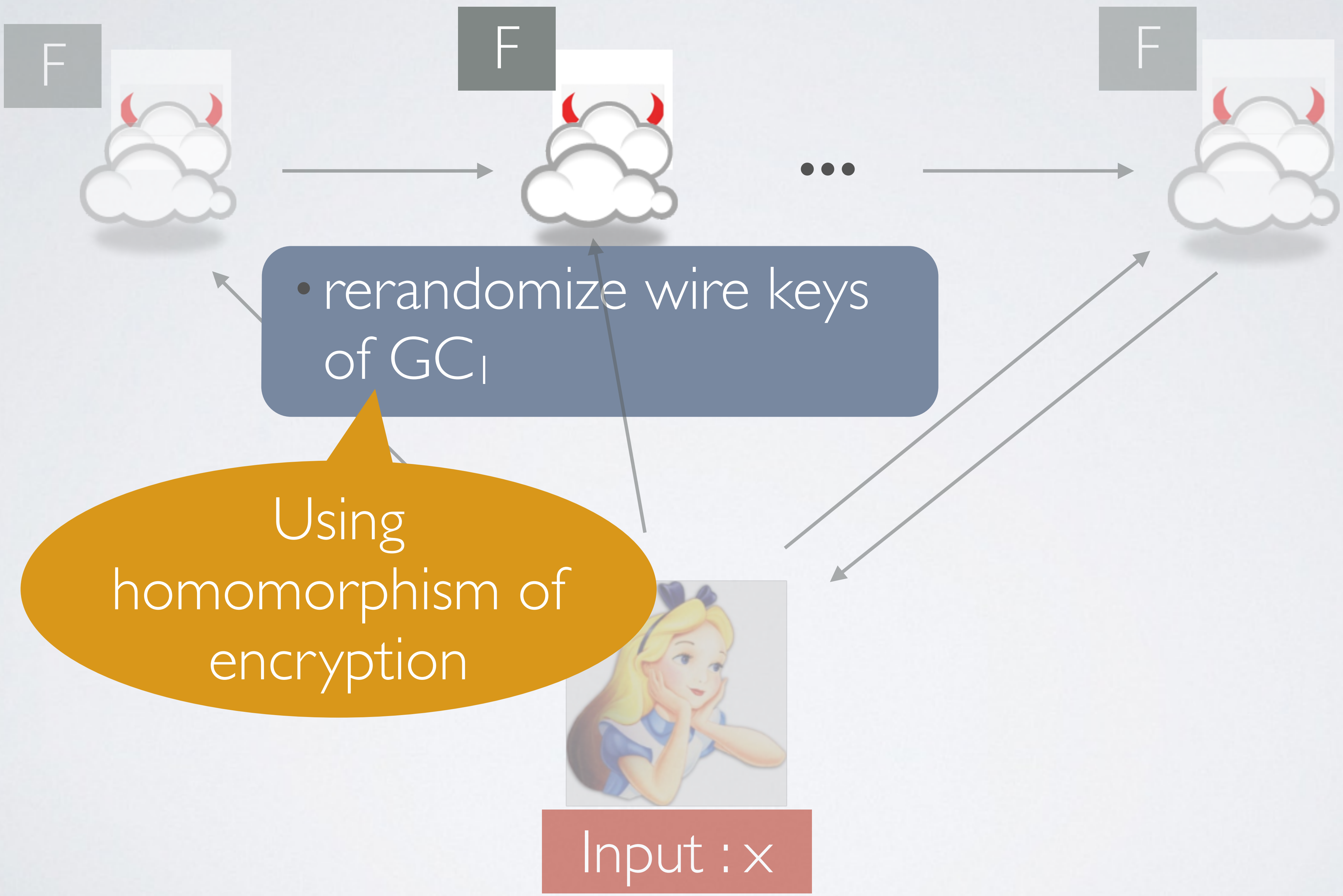
MODIFIED CONSTRUCTION



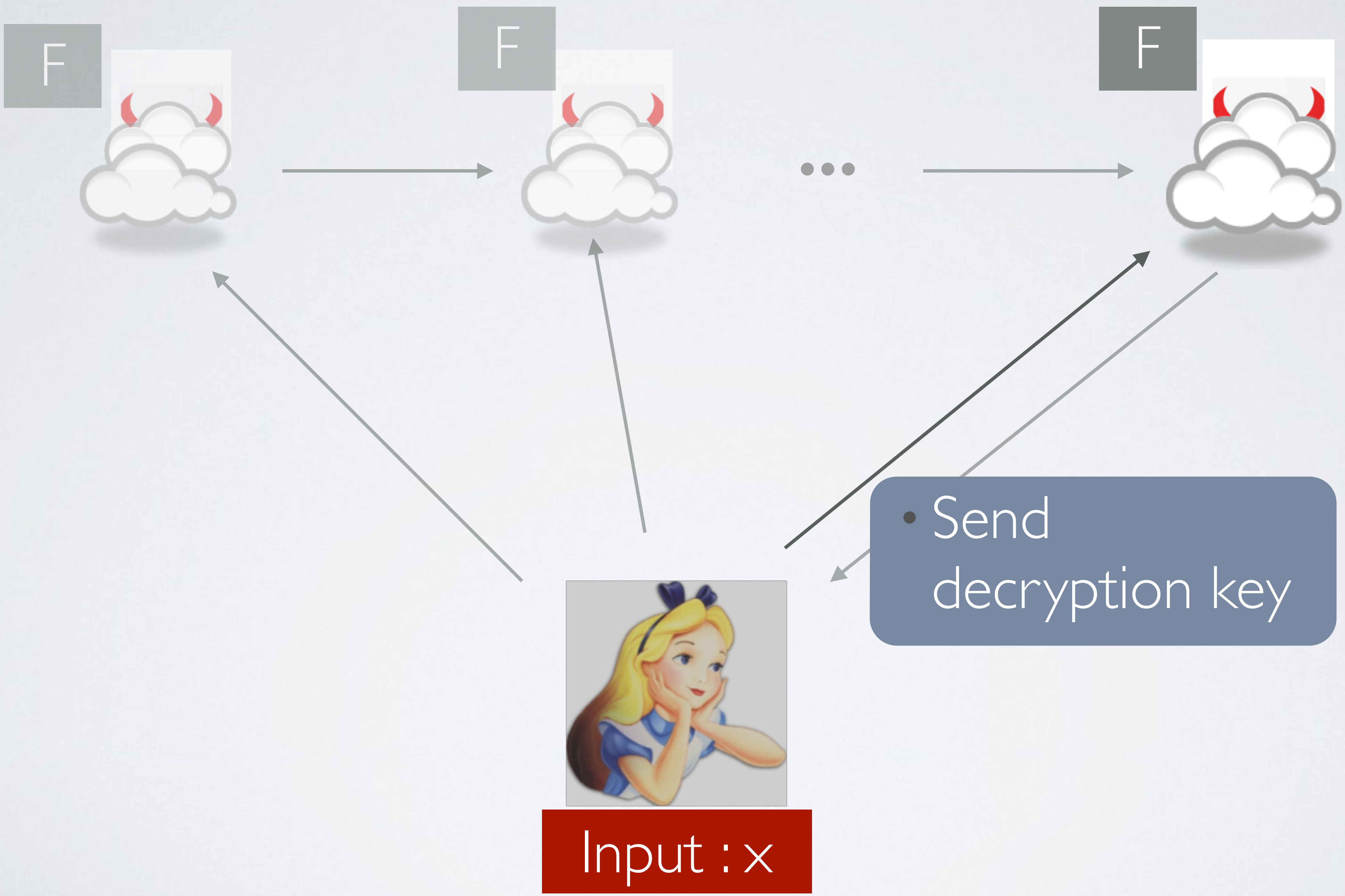
MODIFIED CONSTRUCTION



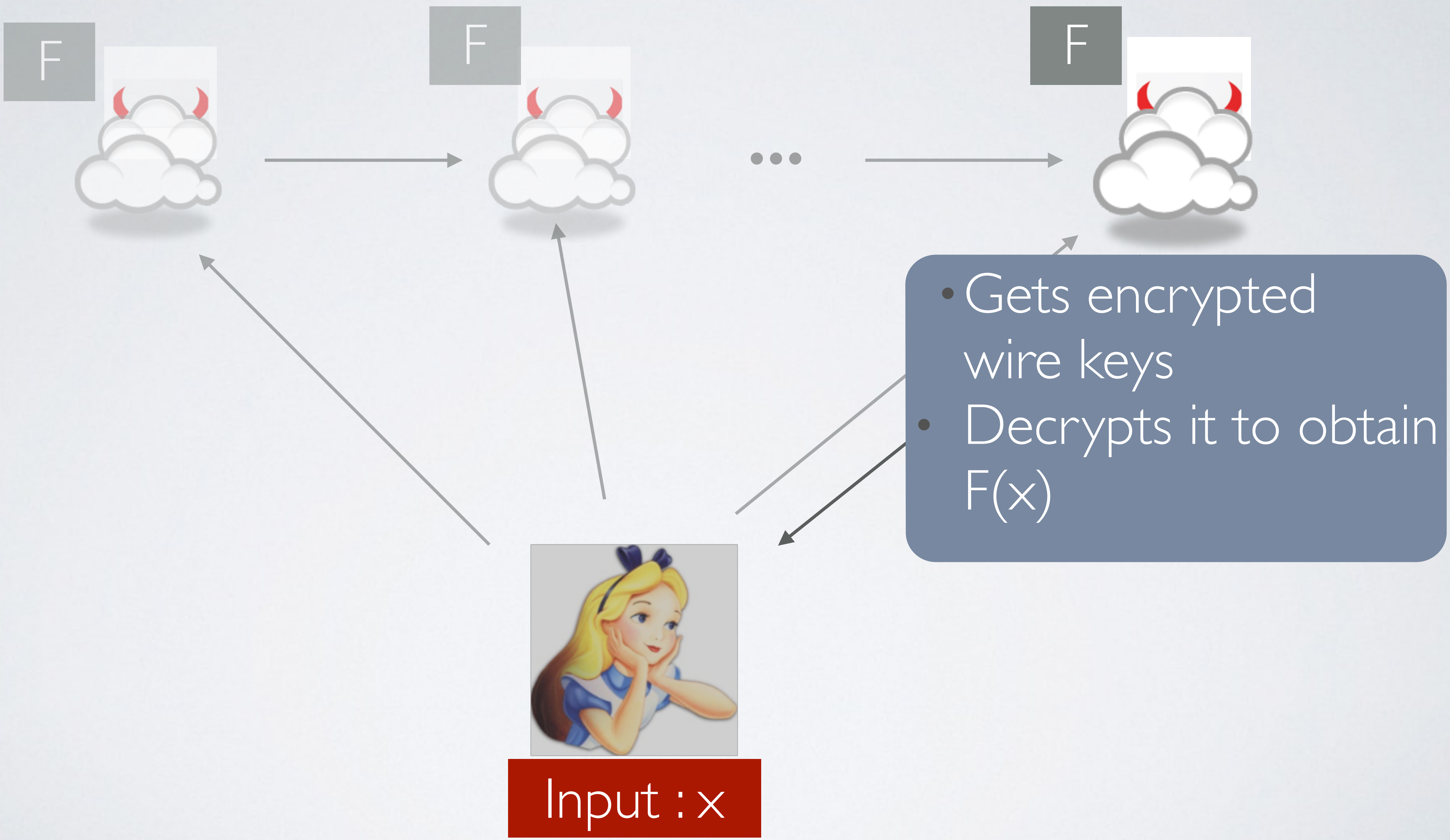
MODIFIED CONSTRUCTION



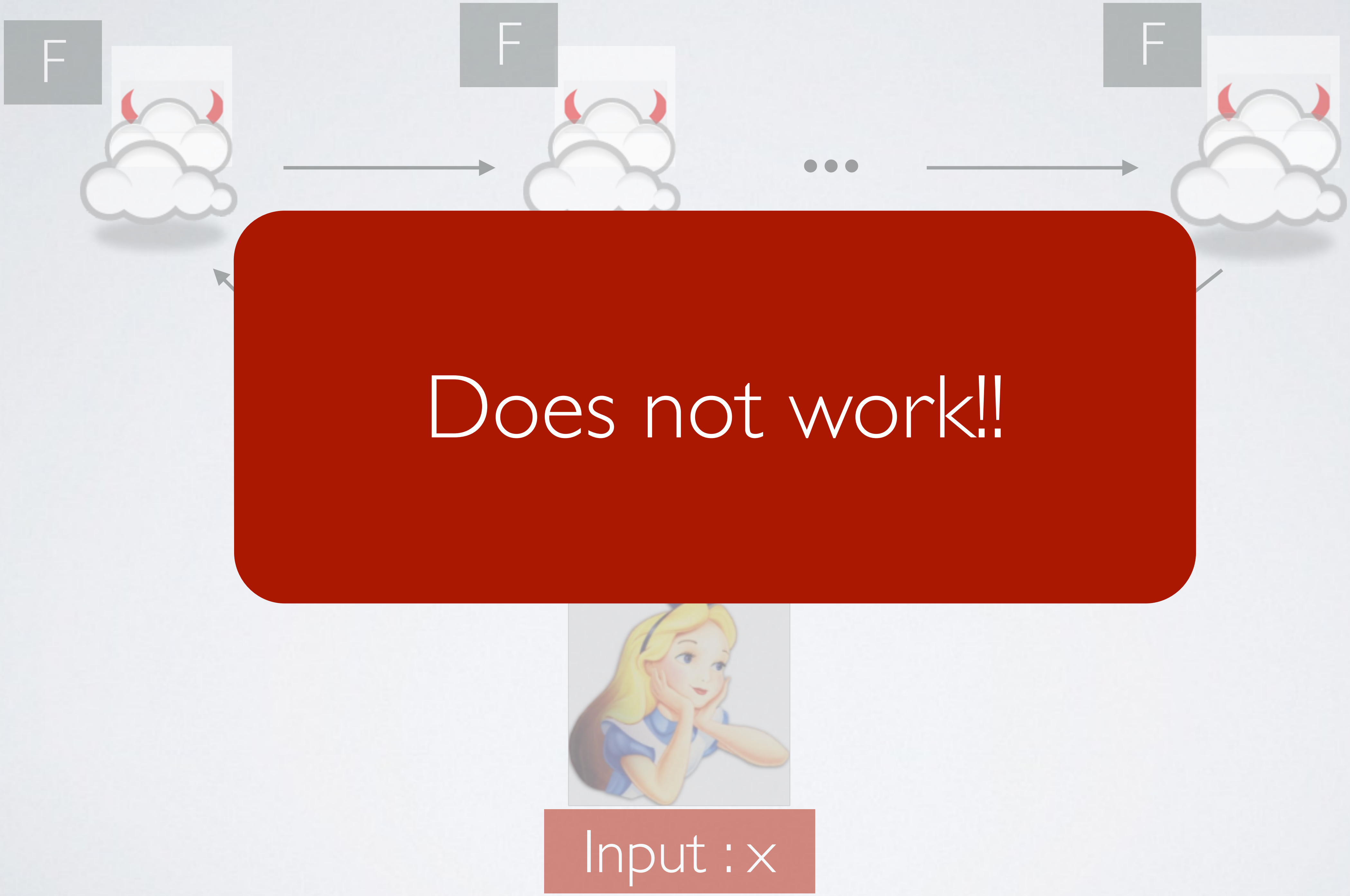
MODIFIED CONSTRUCTION



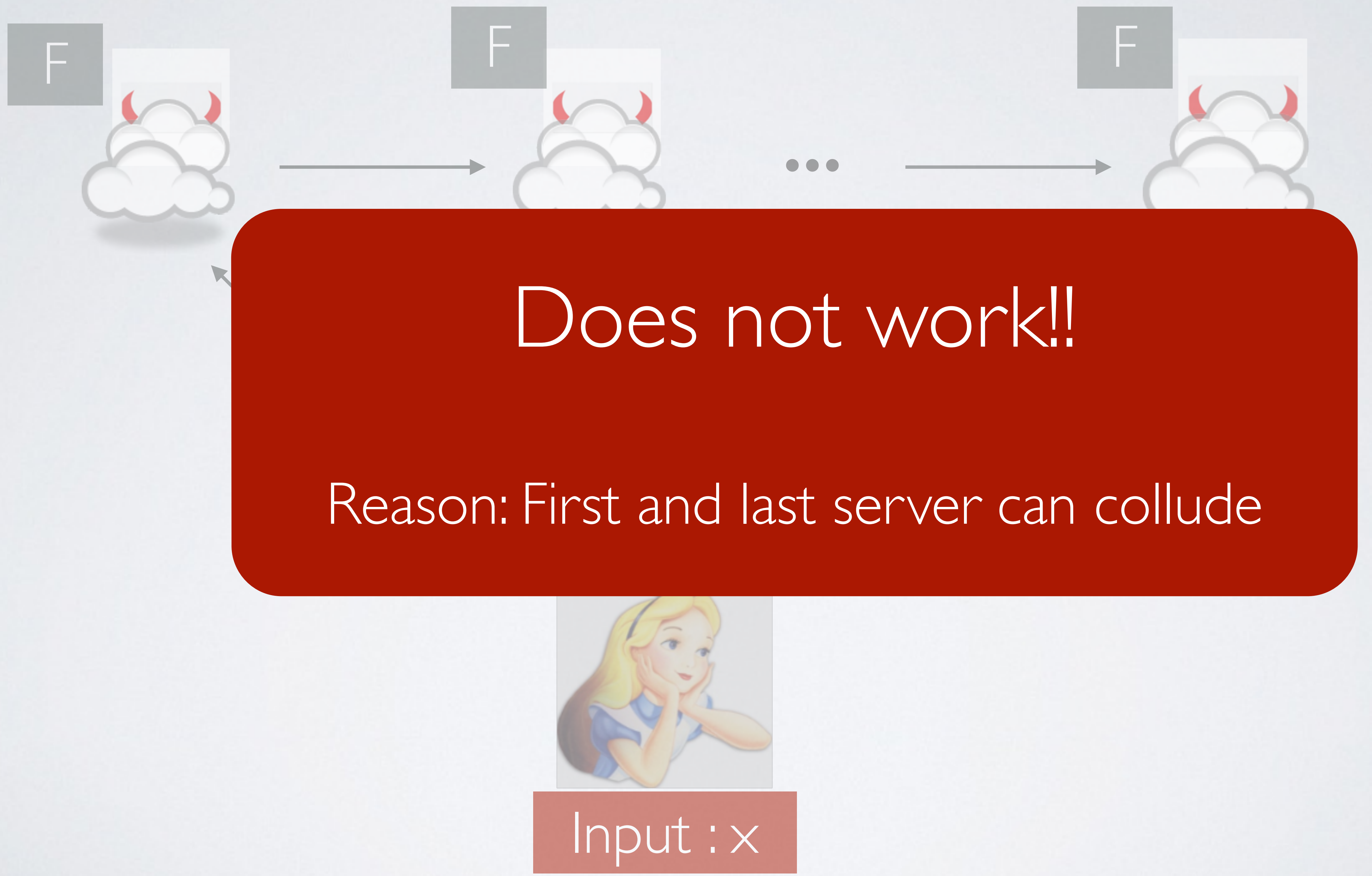
MODIFIED CONSTRUCTION



MODIFIED CONSTRUCTION



MODIFIED CONSTRUCTION



Does not work!!

Reason: First and last server can collude

FIXING THE CONSTRUCTION

- Use re-encryption!

FIXING THE CONSTRUCTION

- Use re-encryption! : can re-encrypt a ciphertext corresponding to a different public key.

FIXING THE CONSTRUCTION

- Use re-encryption! : can re-encrypt a ciphertext corresponding to a different public key.

- We use an encryption scheme that supports re-encryption and homomorphism.

FIXING THE CONSTRUCTION

- Use re-encryption! : can re-encrypt a ciphertext corresponding to a different public key.

Details in the paper!

- We use an encryption scheme that supports re-encryption and homomorphism.

OPEN PROBLEMS

- Replacing NIZKs in our construction.
- VC protocol in the multiple-server model based on one-way functions?
- More efficient VC protocols for specific functions?

QUESTIONS?

Prabhanjan Ananth - prabhanjan@cs.ucla.edu

Nishanth Chandran - nichandr@microsoft.com

Vipul Goyal - vipul@microsoft.com

Bhavana Kanukurthi - bhavanak@cs.bu.edu

Rafail Ostrovsky - rafail@cs.ucla.edu

THANKS!