COSC 544: Course Intro

Justin Thaler

Georgetown University

Logistical Information

1. Instructor: Justin Thaler
2. Email: justin.thaler@georgetown.edu
3. Course webpage: http://people.cs.georgetown.edu/jthaler/COSC544.html

What this course is about

- Different notions of mathematical proofs.
  - And their applications in computer science and cryptography.

- Informally, a proof is anything that convinces someone that a statement is true.

What this course is about

- Different notions of mathematical proofs.
  - And their applications in computer science and cryptography.

- Informally, a proof is anything that convinces someone that a statement is true.
- A "proof system" is specified by a **verification procedure**.
  - The procedure takes as input a statement and "proof" of the statement, and decides whether the statement is valid.
  - The verification procedure tells you what is a convincing proof.

What do we want out of a proof system?

- Any true statement should have a convincing proof.
  - This is called **completeness** of the proof system.
- No false statement should have a convincing proof.
  - This is called **soundness** of the proof system.

What do we want out of a proof system?

- Any true statement should have a convincing proof.
  - This is called **completeness** of the proof system.
- No false statement should have a convincing proof.
  - This is called **soundness** of the proof system.
- The verification procedure should be "efficient".
  - i.e., simple statements should have short proofs that can be **checked** quickly.

What do we want out of a proof system?

- Any true statement should have a convincing proof.
  - This is called **completeness** of the proof system.
- No false statement should have a convincing proof.
  - This is called **soundness** of the proof system.
- The verification procedure should be "efficient".
  - i.e., simple statements should have short proofs that can be **checked** quickly.
- Proving should be efficient too.
  - i.e., if a prover knows "why" a statement is true, it should not require much work for the prover to generate a convincing proof.

Historical Context

- Traditionally, a mathematical proof is something that can be **written down** and checked step-by-step for correctness.
  - Each step should either be trivial to verify, or else false.
  - This has been the de facto notion of proof since roughly 600 BCE (developed by ancient Greek mathematicians).

Historical Context

- Traditionally, a mathematical proof is something that can be **written down** and checked step-by-step for correctness.
  - Each step should either be trivial to verify, or else false.
  - This has been the de facto notion of proof since roughly 600 BCE (developed by ancient Greek mathematicians).
  - In computer science, this traditional notion corresponds to the complexity class **NP**.

Historical Context

- Since 1985, computer scientists have studied much more general/exotic notions of proofs.
  - This has transformed our notion of what it means to prove something, and led to major advances in cryptography.

What Kinds of Non-Traditional Notions of Proofs Will We Study?

What notions of proof systems will we study?

- All notions in this course will be **probabilistic**.
  - The verification procedure will make random choices.
  - And there will be a very small probability of declaring a "false" statement and proof to be valid.

What notions of proof systems will we study?

- All notions in this course will be **probabilistic**.
  - The verification procedure will make random choices.
  - And there will be a very small probability of declaring a "false" statement and proof to be valid.

- Interactive proofs (IPs)
- Argument systems
- Zero-knowledge IPs and arguments
- Multi-prover interactive proofs (MIPs)
- Probabilistically checkable proofs (PCPs)
- etc.

Interactive Proofs (IPs)

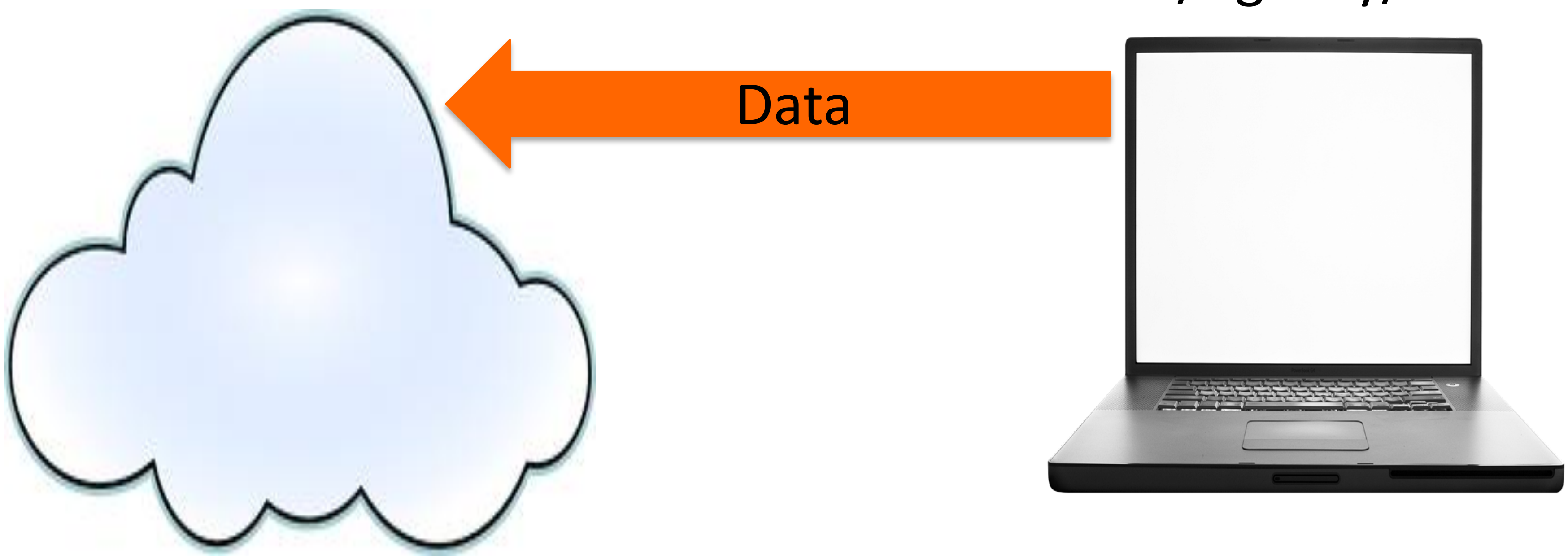
Cloud Provider

Business/Agency/Scientist

Interactive Proofs (IPs)

Cloud Provider

Business/Agency/Scientist

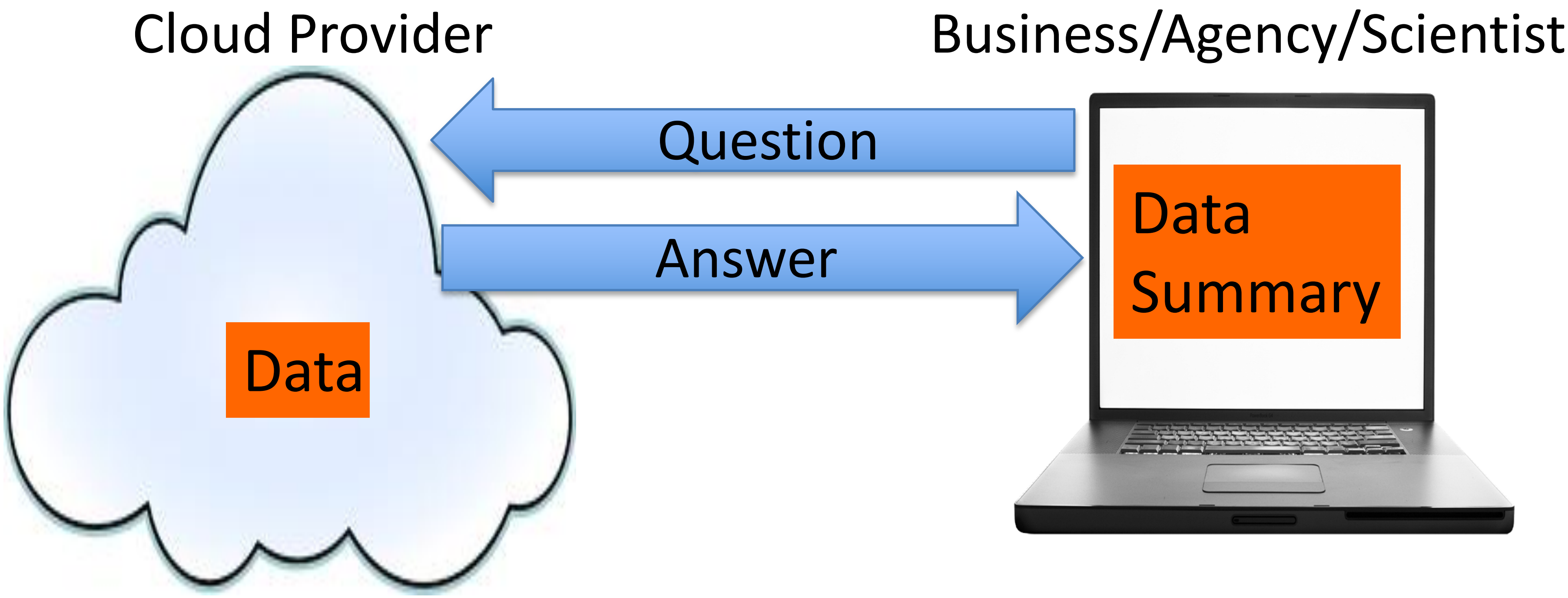


Interactive Proofs (IPs)

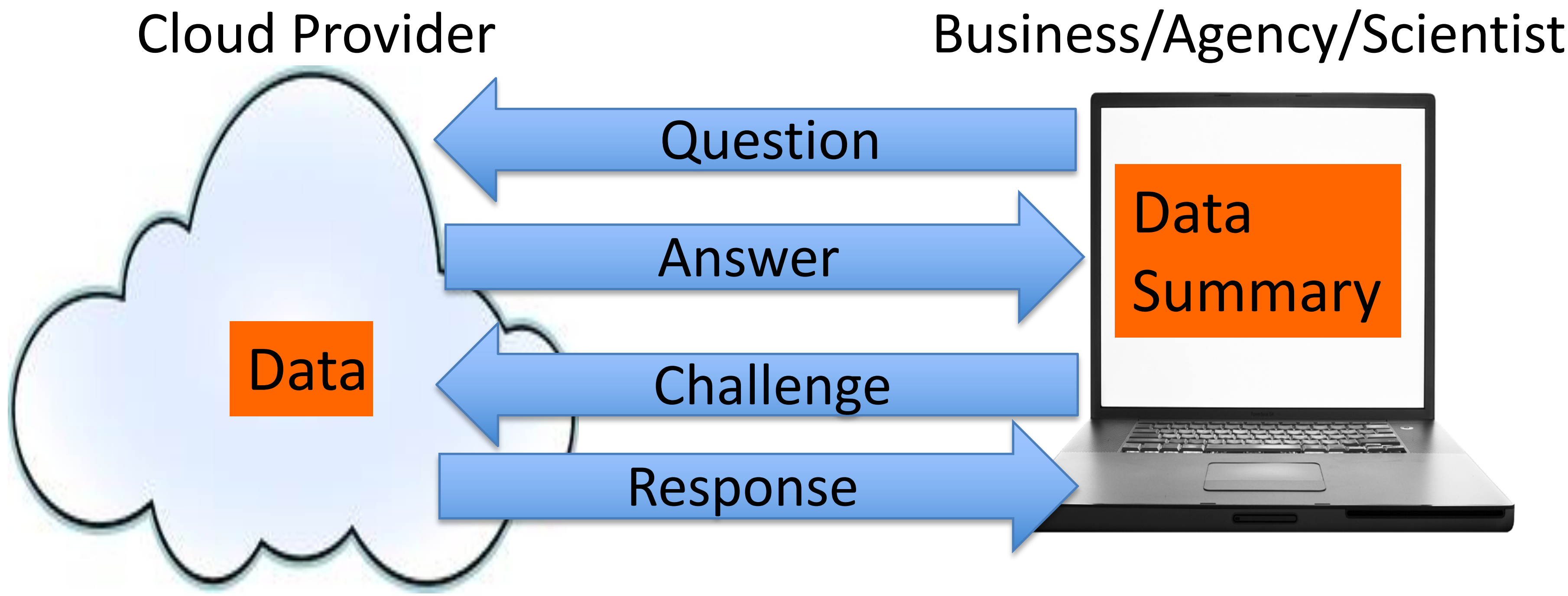
Cloud Provider

Business/Agency/Scientist

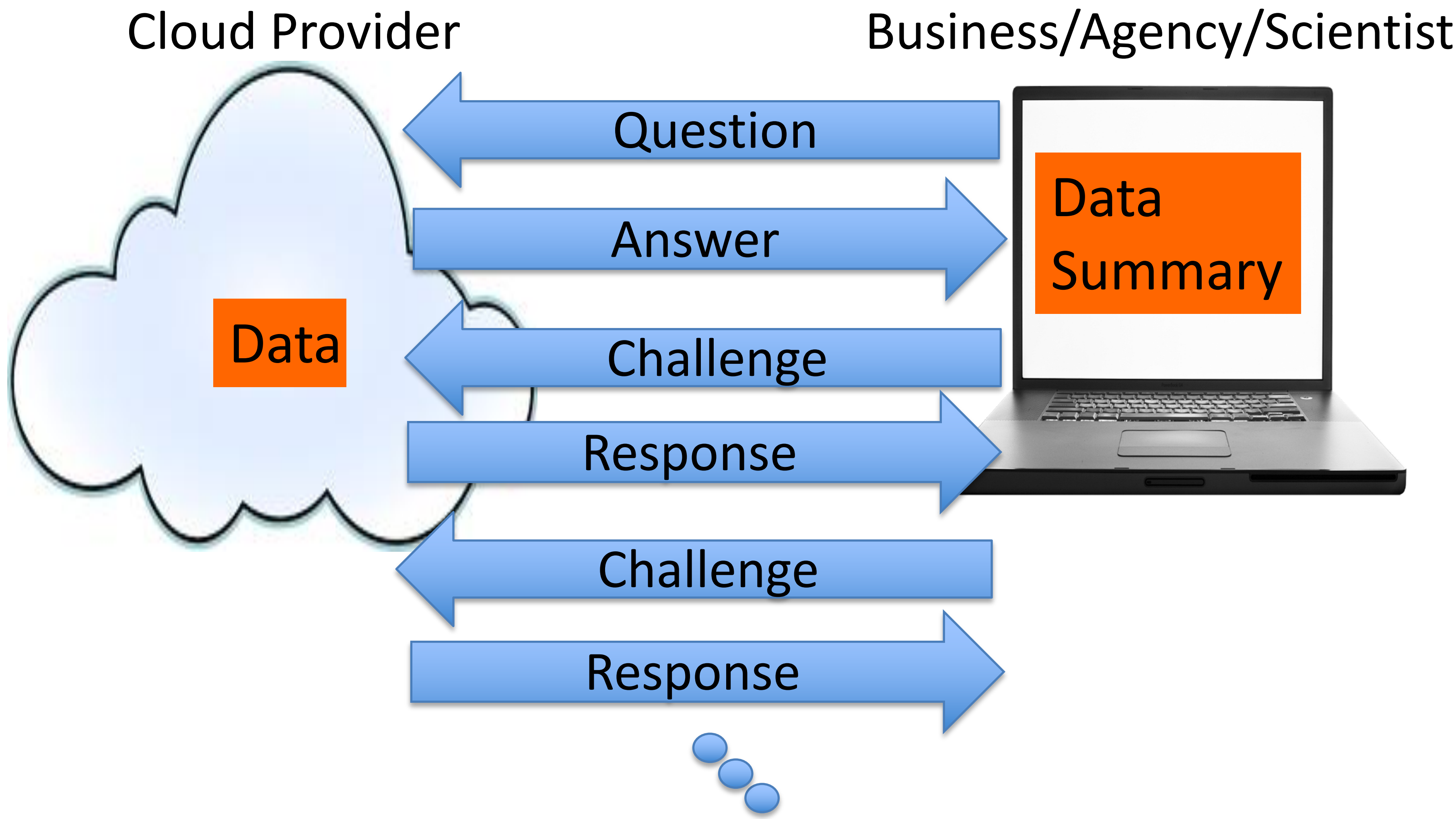
Interactive Proofs (IPs)



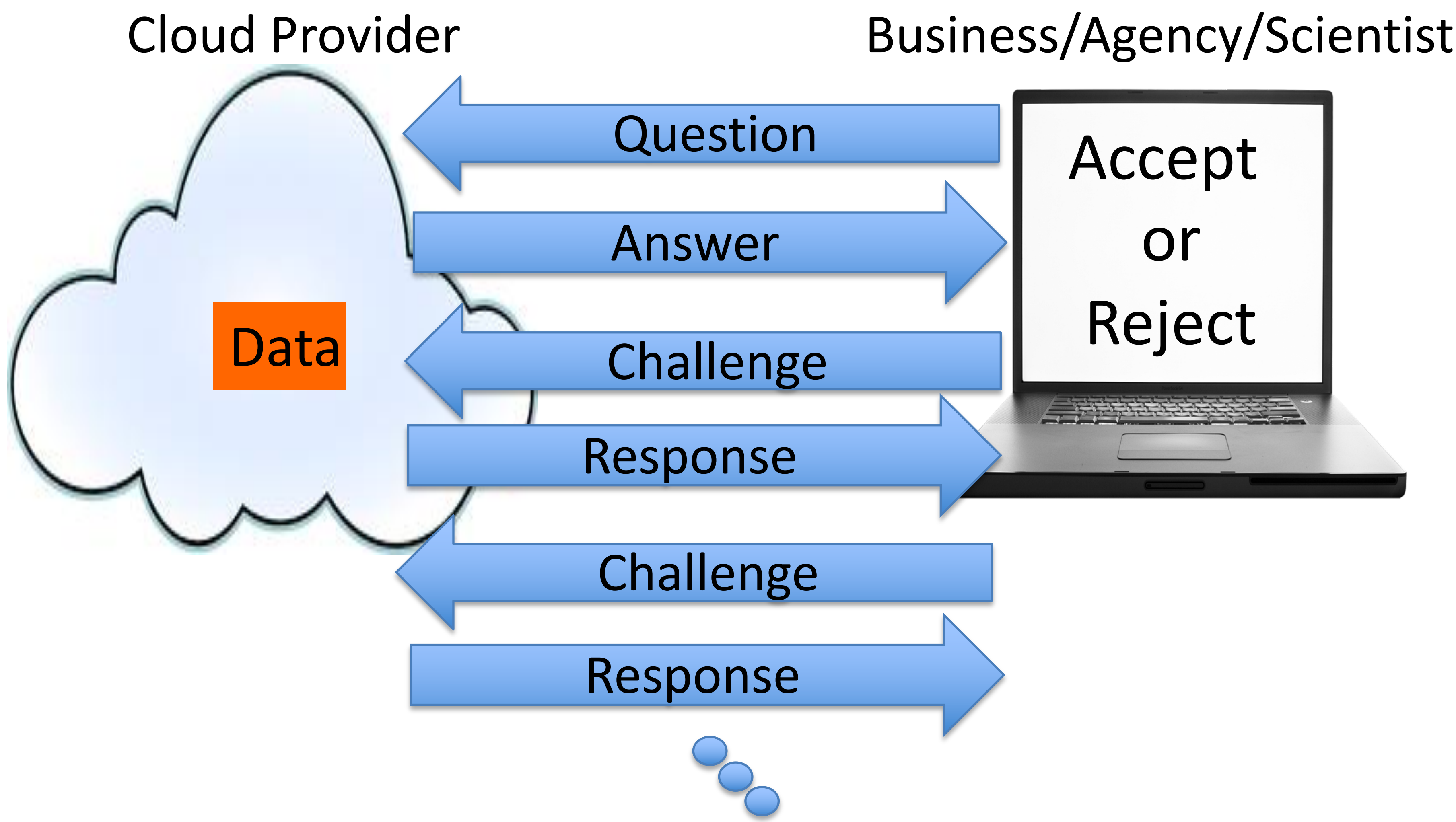
Interactive Proofs (IPs)



Interactive Proofs (IPs)



Interactive Proofs (IPs)



Interactive Proofs

- Prover P and Verifier V.

- P solves problem, tells V the answer.
  - Then P and V have a conversation.
  - P's goal: convince V the answer is correct.

- Requirements:
  - 1. Completeness: an honest P can convince V to accept.
  - 2. Soundness: V will catch a lying P with high probability.

Interactive Proofs

- Prover P and Verifier V.

- P solves problem, tells V the answer.
  - Then P and V have a conversation.
  - P's goal: convince V the answer is correct.

- Requirements:
  - 1. Completeness: an honest P can convince V to accept.
  - 2. Soundness: V will catch a lying P with high probability.
    - This must hold even if P is computationally unbounded and trying to trick V into accepting the incorrect answer.

Argument Systems

- Same as IPs, except soundness holds only against cheating provers that run in polynomial time.

Argument Systems

- Same as IPs, except soundness holds only against cheating provers that run in polynomial time.
  - Argument systems make use of cryptosystems.
  - If P can "break" the cryptosystem, then P can convince V of false statements. But breaking a cryptosystem is assumed to require vast computational resources (superpolynomial time).

Argument Systems

- Same as IPs, except soundness holds only against cheating provers that run in polynomial time.
  - Argument systems make use of cryptosystems.
  - If P can "break" the cryptosystem, then P can convince V of false statements. But breaking a cryptosystem is assumed to require vast computational resources (superpolynomial time).
- IPs were introduced by [Goldwasser, Micali, Rackoff 1985] and [Babai 1985].
- Argument systems were introduced by [Brassard, Chaum, Crepeau 1988].

Zero-Knowledge Proofs and Arguments

- These are proofs that reveal nothing to the verifier other than the validity of the statement being proven.
    - They have many applications in cryptography.

Example Application: Authentication

- Example: authentication.
  - Suppose Alice chooses a random password x and publishes a hash $z = h(x)$, where h is a **one-way hash function**.
    - h is **"easy to compute, but hard to invert".**
    - For a random x, given only $z = h(x)$, it is **hard** to find a preimage x' of z under h.

Example Application: Authentication

- Example: authentication.
  - Suppose Alice chooses a random password x and publishes a hash $z = h(x)$, where h is a **one-way hash function**.
    - h is **"easy to compute, but hard to invert".**
    - For a random x, given only $z = h(x)$, it is **hard** to find a preimage x' of z under h.
  - Later, Alice wants to convince Bob that she is the same person who published z.

Example Application: Authentication

- Example: authentication.
  - Suppose Alice chooses a random password x and publishes a hash $z = h(x)$, where h is a **one-way hash function**.
    - h is **"easy to compute, but hard to invert".**
    - For a random x, given only $z = h(x)$, it is **hard** to find a preimage x' of z under h.
  - Later, Alice wants to convince Bob that she is the same person who published z.
  - Alice can do this by proving to Bob that she knows an x' such that $h(x') = z$.
  - This convinces Bob that either Alice know x to begin with, or else she inverted h, which is assumed to be beyond anyone's capabilities.

Example Application: Authentication

- How can Alice prove to Bob that she knows an x' such that $h(x') = z$?
  - Obvious approach: Alice can just send x' to Bob.
  - But this reveals x' to Bob!
  - From then on, Bob can impersonate Alice, as he also knows an x' such that $h(x') = z$.

Example Application: Authentication

- How can Alice prove to Bob that she knows an x' such that $h(x') = z$?
  - Obvious approach: Alice can just send x' to Bob.
  - But this reveals x' to Bob!
  - From then on, Bob can impersonate Alice, as he also knows an x' such that $h(x') = z$.
- In order to prevent Bob from learning any information that might help him compromise the password, it is essential that the proof reveal nothing other than that Alice knows an x' such that $h(x') = z$.
  - This is exactly what a zero-knowledge proof guarantees.

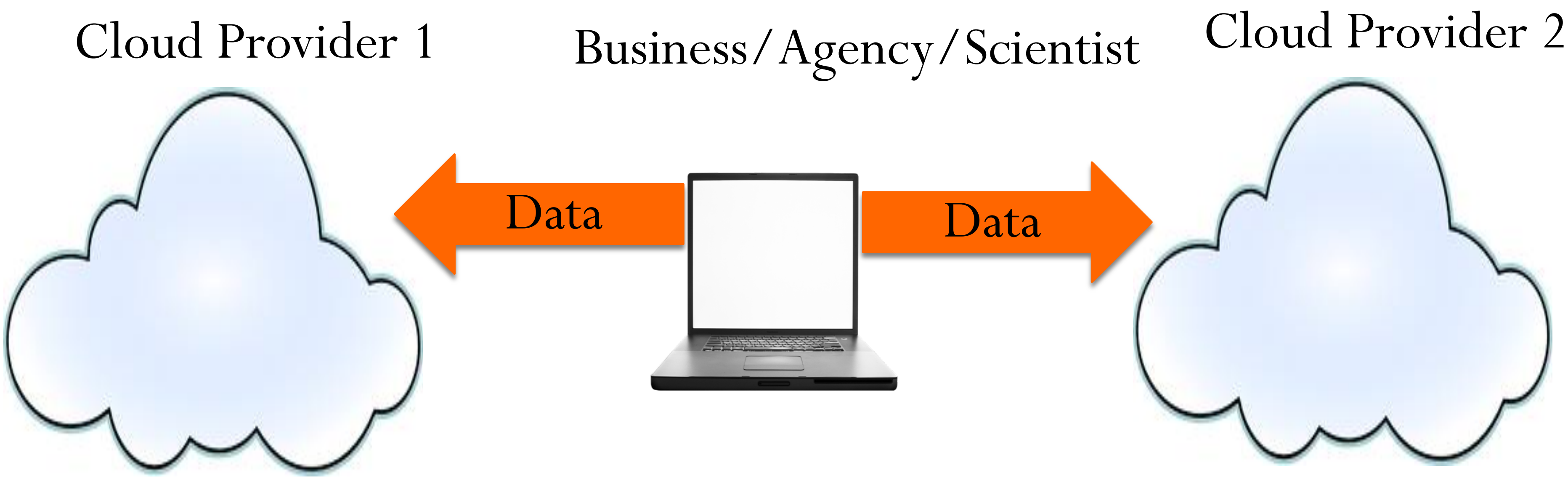
Multi-Prover Interactive Proofs (IPs)

Cloud Provider 1

Business/Agency/Scientist

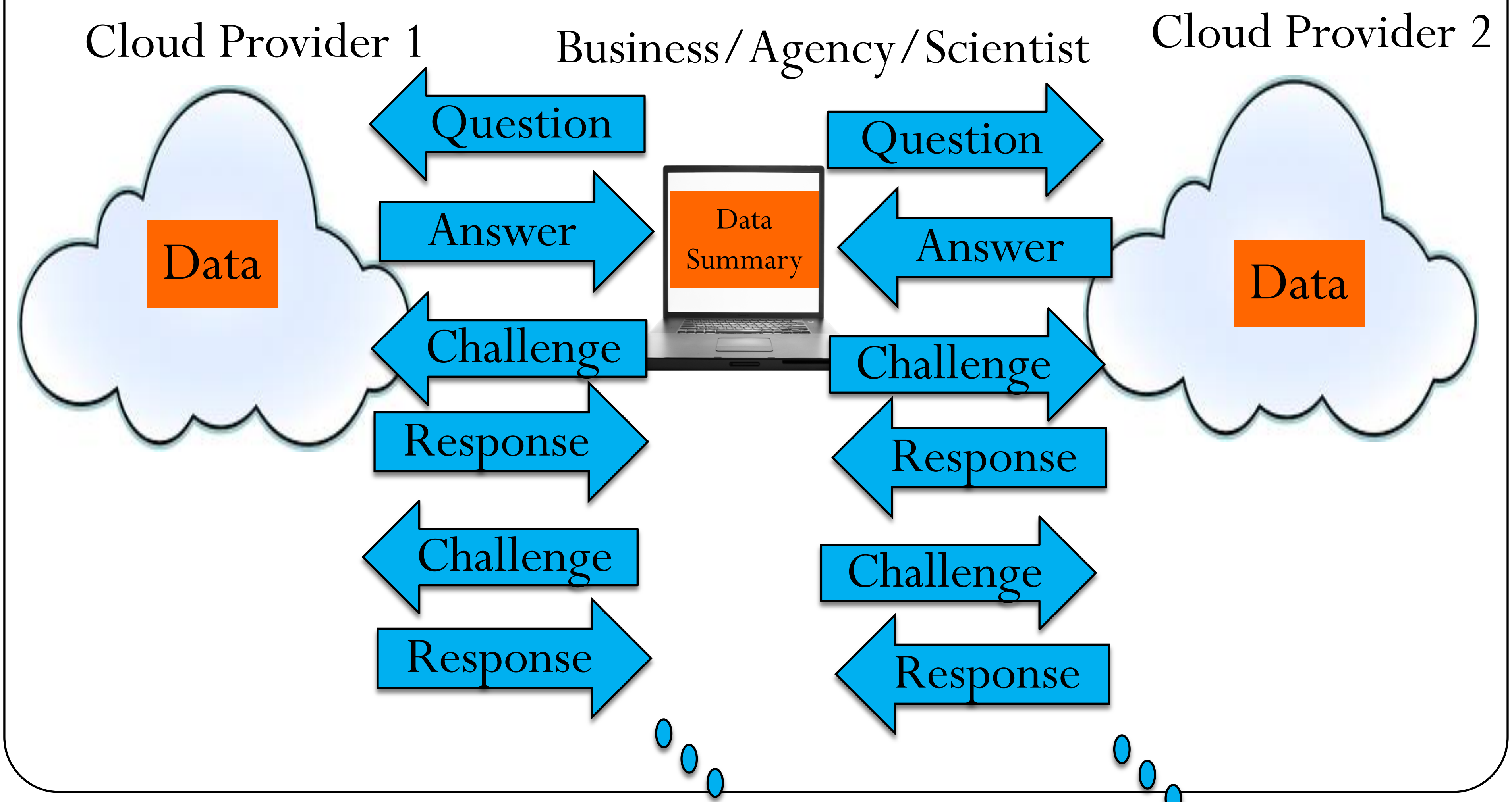
Cloud Provider 2

Multi-Prover Interactive Proofs (IPs)

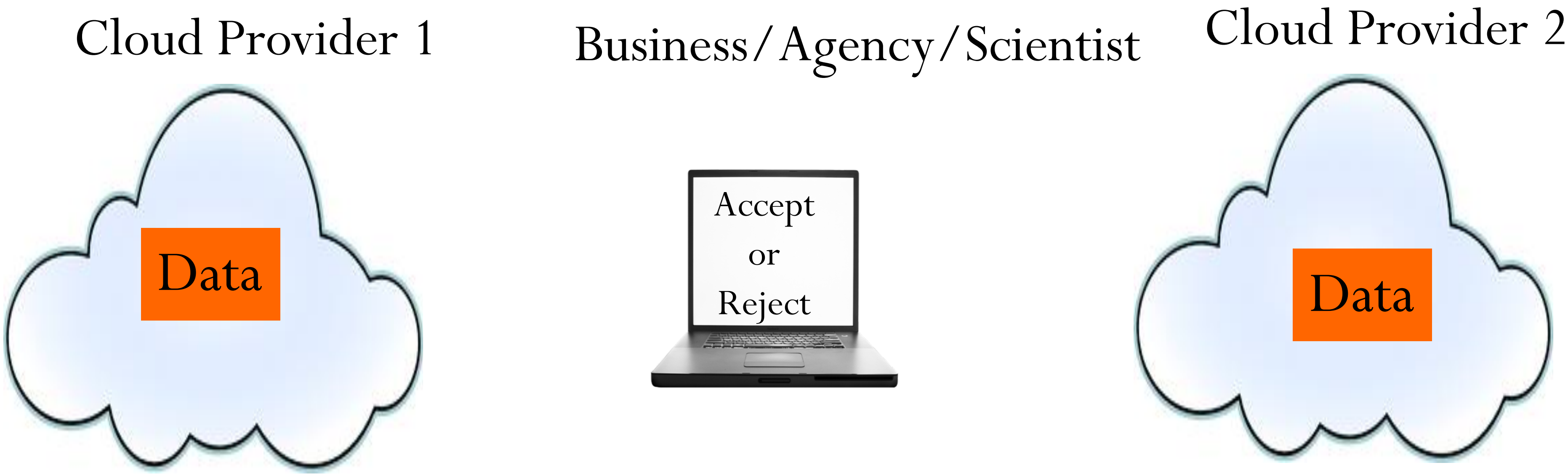


Multi-Prover Interactive Proofs (IPs)

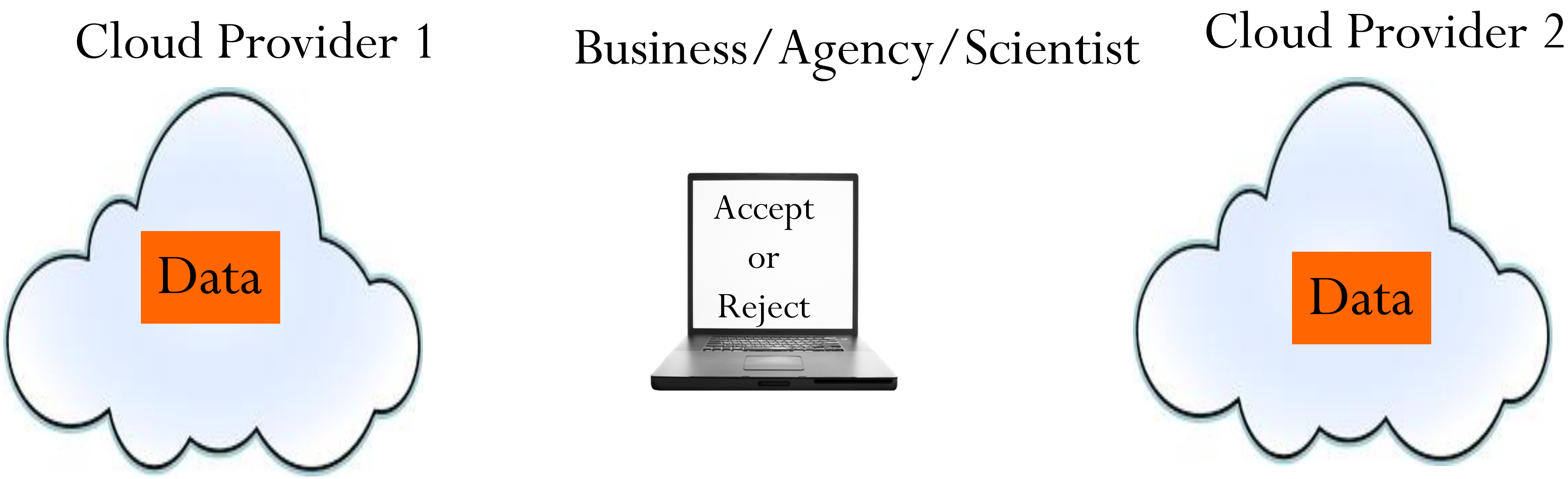
Multi-Prover Interactive Proofs (IPs)



Multi-Prover Interactive Proofs (IPs)



Multi-Prover Interactive Proofs (IPs)



Key assumption of the model: Cloud Provider 1 does not inform Cloud Provider 2 of the challenges it receives, and vice versa.

Probabilistically Checkable Proofs (PCPs)

- A classic, static proof , but the verifier only looks at a few symbols of the proof.

Context for this course

- Theorists showed in the 1980s and 1990s that IPs and arguments can be vastly more efficient (asymptotically) than traditional static proofs.
  - i.e., far more complicated statements can be verified efficiently using these exotic notions of proofs, compared to static proofs.

Context for this course

- Theorists showed in the 1980s and 1990s that IPs and arguments can be vastly more efficient (asymptotically) than traditional static proofs.
  - i.e., far more complicated statements can be verified efficiently using these exotic notions of proofs, compared to static proofs.
- Yet these results were considered wildly impractical.
  - Generating proofs for even very simple statements would have taken **trillions** of years in practice.

Context for this course

- Theorists showed in the 1980s and 1990s that IPs and arguments can be vastly more efficient (asymptotically) than traditional static proofs.
  - i.e., far more complicated statements can be verified efficiently using these exotic notions of proofs, compared to static proofs.
- Yet these results were considered wildly impractical.
  - Generating proofs for even very simple statements would have taken **trillions** of years in practice.
  - But the last decade has seen major improvements in the costs of these exotic proof systems.
  - They have seen deployment in commercial settings.

Context for this course

- Most useful in practice are zero-knowledge arguments.
- We are mainly studying IPs, MIPs, and PCPs because they are "building blocks" used for designing zero-knowledge arguments.

Why you should take this course

- It’s cool. It may change how you think about what it means to prove something or be convinced that something is true.

Why you should take this course

- It's cool. It may change how you think about what it means to prove something or be convinced that something is true.
- You'll learn some of the most celebrated results and techniques in computer science and cryptography.
  - Cryptographic tools include:
    - Collision-resistant hash functions.
    - Commitment schemes.
    - Homomorphic encryption.
    - Pairing-based cryptography.

Why you should take this course

- It's cool. It may change how you think about what it means to prove something or be convinced that something is true.
- You'll learn some of the most celebrated results and techniques in computer science and cryptography.
  - Cryptographic tools include:
    - Collision-resistant hash functions.
    - Commitment schemes.
    - Homomorphic encryption.
    - Pairing-based cryptography.
- You'll be on the cutting edge of a major research area.
  - Which may play a role in transforming basic societal functions in the coming decades.
  - E.g., asset transfer, identification, licensing, etc.