电商金融技术

华东理工大学计算机系
翟洁

电商金融技术

- 1、电商金融安全简介
- 2、云计算简介
- 3、大模型简介

1.1 加密算法

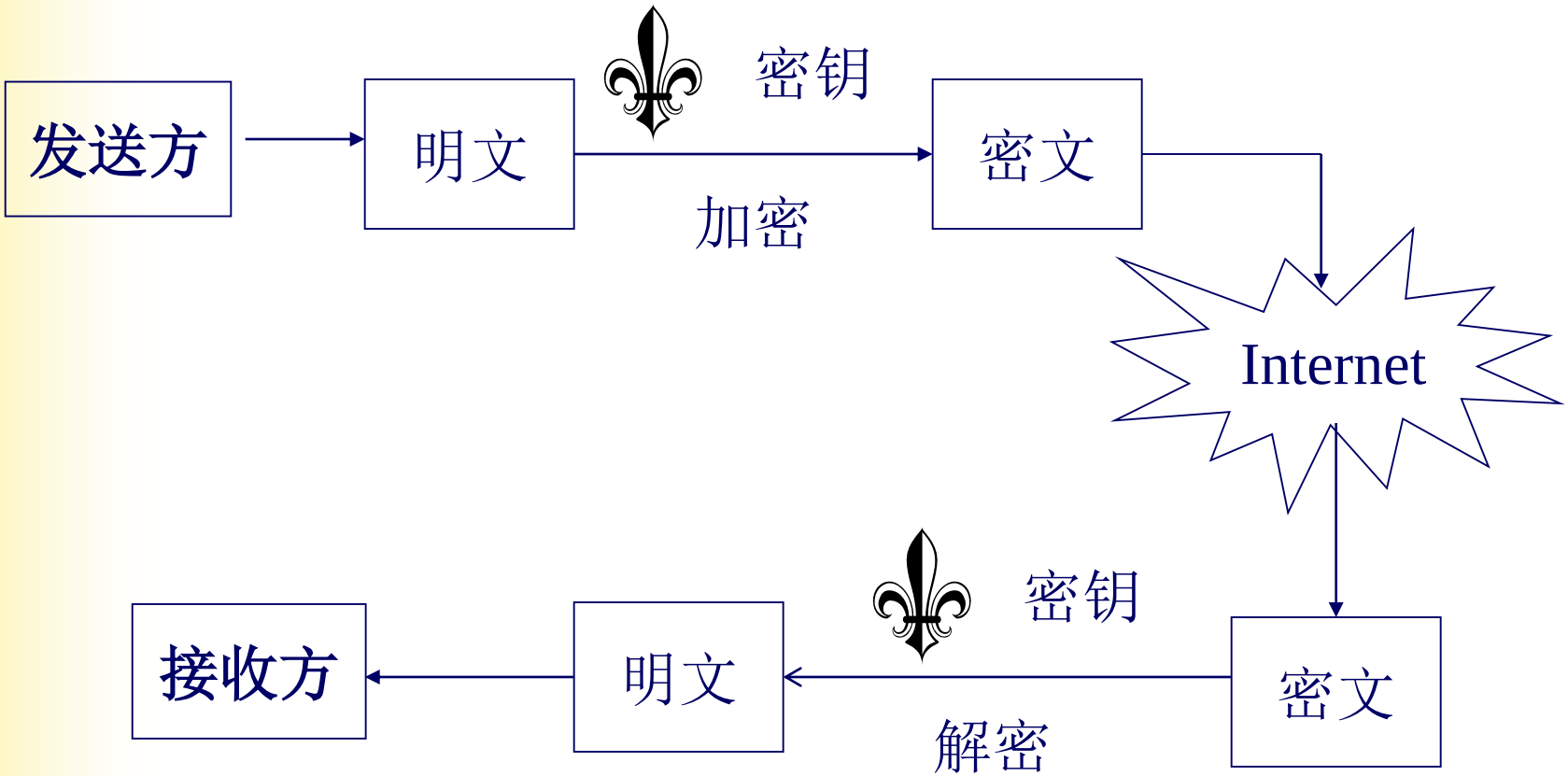
1.2 数字签名

1.3 支付协议

加密算法

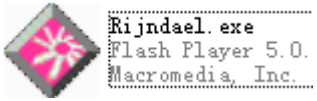
对称加密体制

——加密和解密使用同一把密钥



DES不足：主要是DES的密钥长度只有56位，不能抵抗穷举密钥搜索攻击。

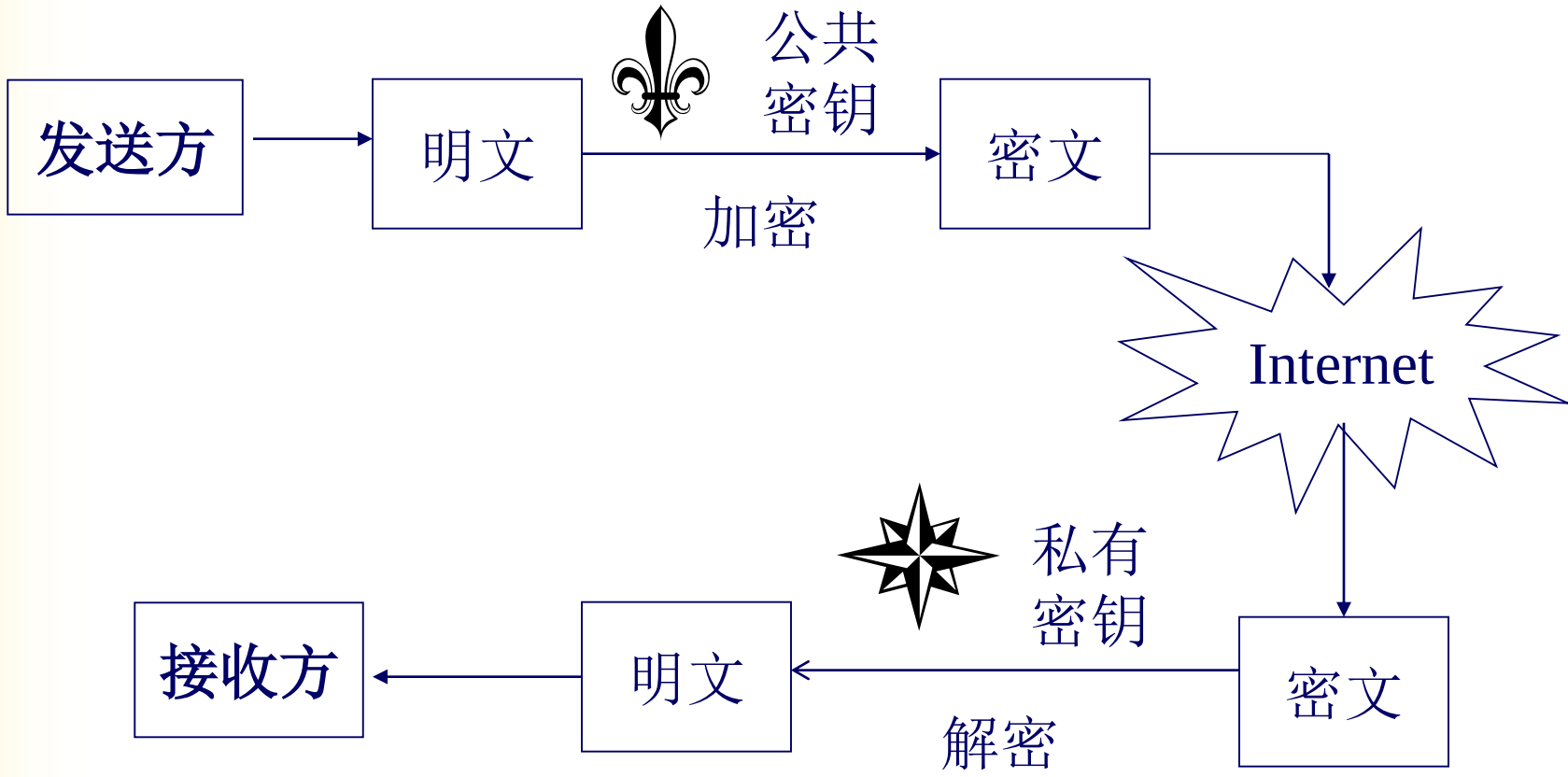
AES加密算法采用分组密码体制，每个分组数据的长度为128位16个字节，密钥长度可以是128位16个字节、192位或256位，一共有四种加密模式。



加密算法

不对称加密体制

——加密和解密使用不同的密钥



特点：安全性好，但速度较慢

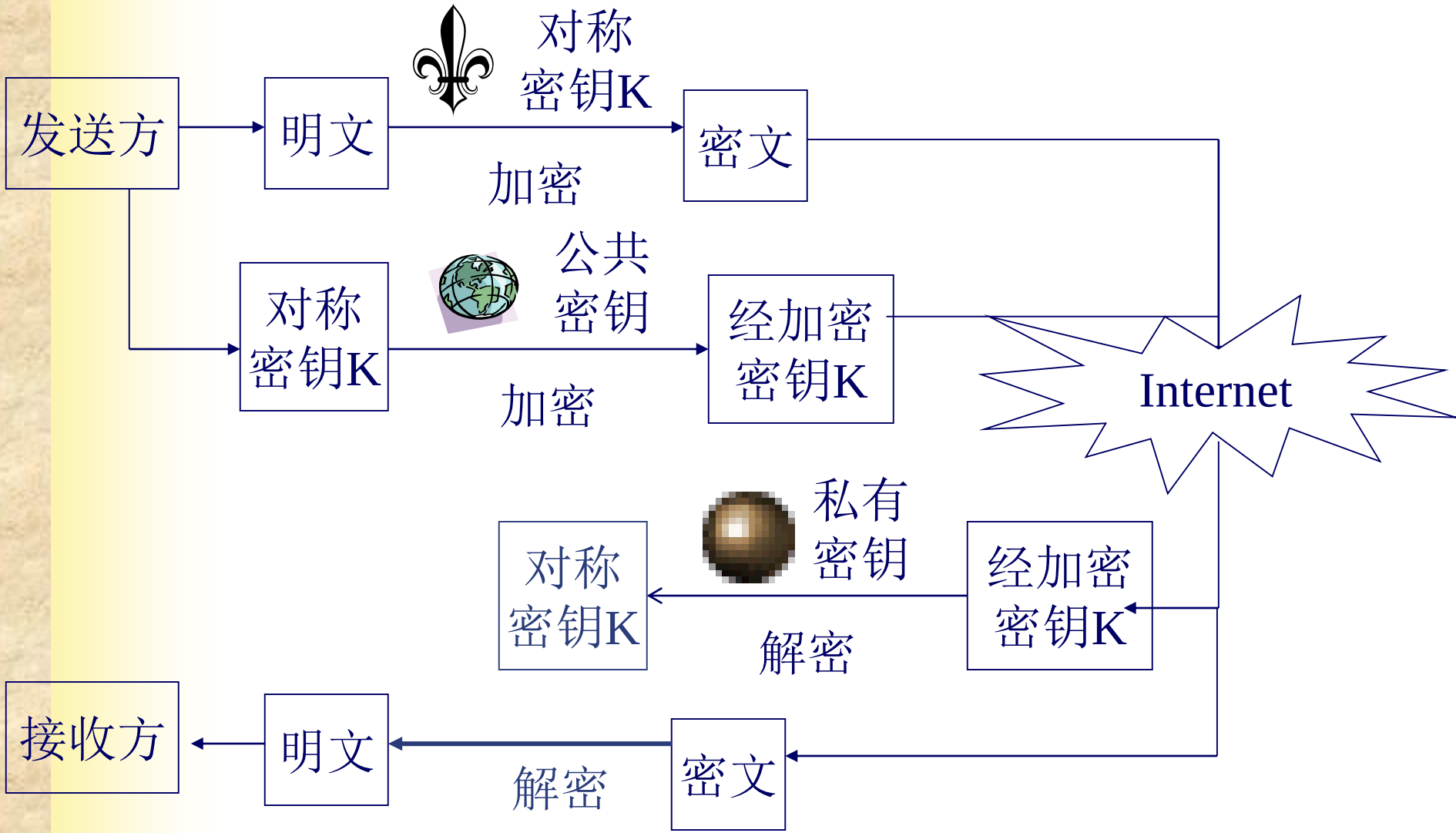
不对称密钥

- 02是原文；128是密文
- $C1 = m1^{e} = 02^{97} = 128 \ (\mathrm{mod}\ 209)$
- （209，97）是公钥

- $M1 = C1^{d} = 128^{13} = 02 \ (\mathrm{mod}\ 209)$
- （209，13）是私钥

加密方法

数字信封：对称加密和不对称加密相结合



每次交换信息都生成一把对称密钥

1.1 加密算法

1.2 数字签名

1.3 支付协议

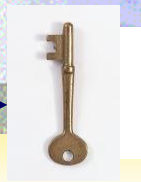
数字签名

散列函数

——一种可以产生一个称为消息摘要的固定长度数字的算法。

01011001101 128位

散列摘要



发送方私钥加密
(生成数字签名)

散列
函数

Buy xyz@$52

对称密钥加密
（生成密文）

对收到的明文实施散列函数重新得到散列值。

比较解密的散列值和重新计算获得的散列值。

若一致则表示未被破坏。

接收方公钥加密
（生成加密的
对称密钥）

internet

01011001101 128位

发送方公钥
（解密得到摘要）

散列摘要

验证消息真实性

接收方

Buy xyz@$52

对称密钥
（解密得到明文）

接收方私钥
（解密得到对称密钥）

原来的订单

Buy xyz@$52

散列摘要

解密后的订单

Buy xzy@$52

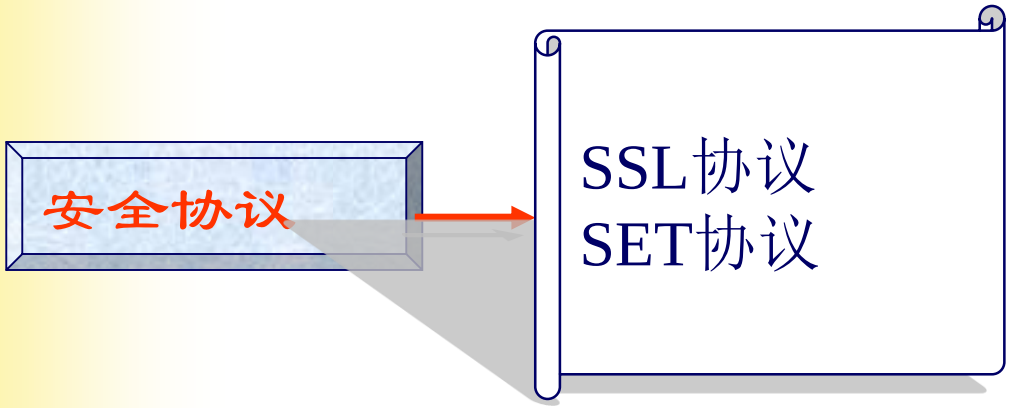
散列摘要

不一致，所以信息的完整性被破坏

1.1 加密算法

1.2 数字签名

1.3 支付协议


安全协议
SSL协议
SET协议

SSL（Secure Sockets Layer）协议

客户
浏览器

商家
服务器

双方的数字证书

安全交易过程

打印交易数据
留作凭证

客户

银行

开始
付货

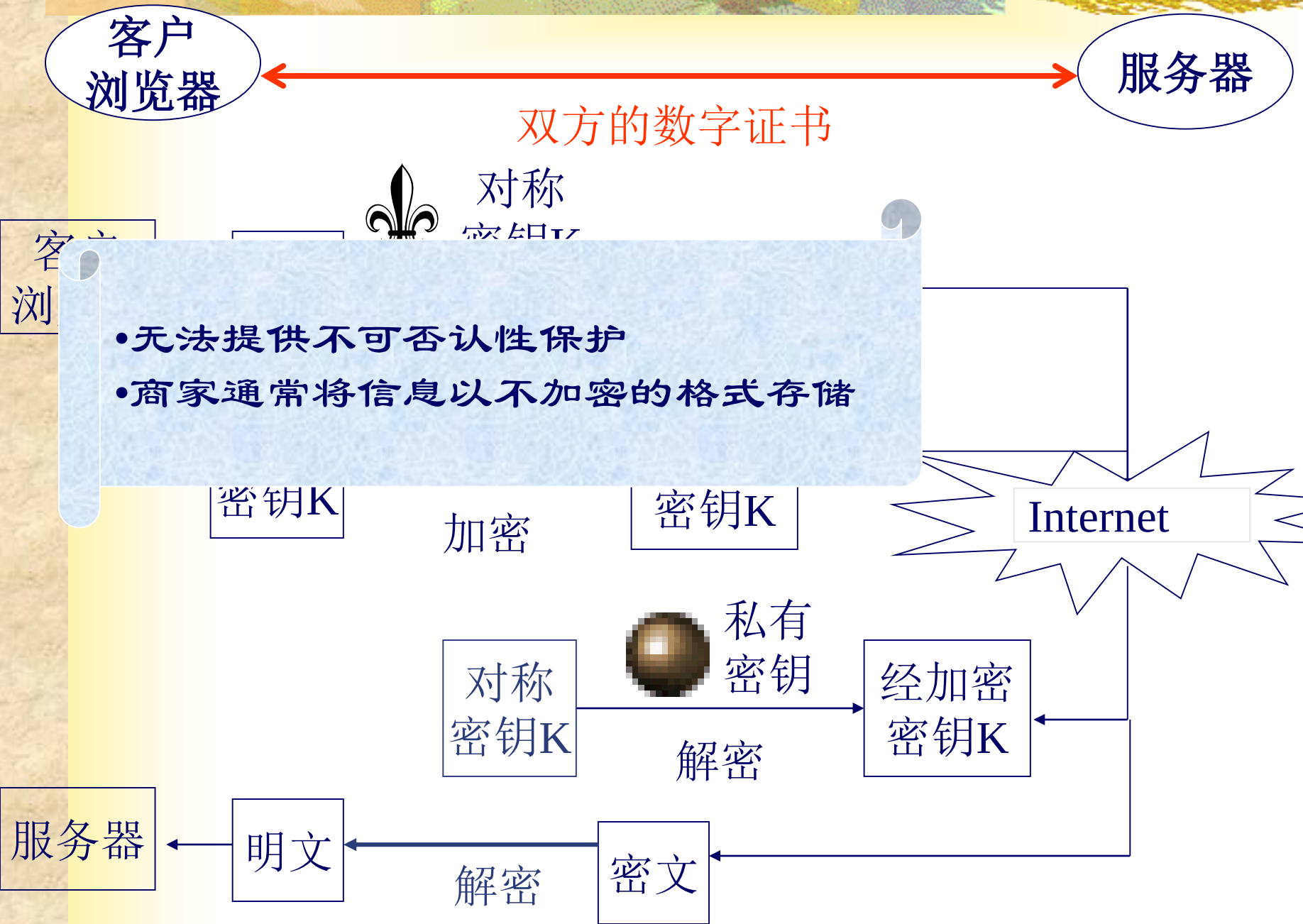
确认或
付款成功

采购
订单
付款
信息

付款
信息

商家

SSL（Secure Sockets Layer）协议



支付协议

SET（Secure Electronic Transaction）协议

双向签名

- 订购指令→商户
- 付款指令→支付网关

缺点：

在银行网络、商家服务器、顾客的PC上安装相应的软件，所以价格昂贵。

商家

通知商家发货

商家的软件转发

加密信息

清算所

商家银行

相互验证身份

SET加密得到的

订单和支付信息发送

清算所向发卡行

审核帐户和余额

发卡行将

款项划拨到

商家银行

消费者

消费者发卡行

选定SET支付

数字钱包、证书

电商金融安全简介小结

- 加密技术
- 数字签名
- 支付协议

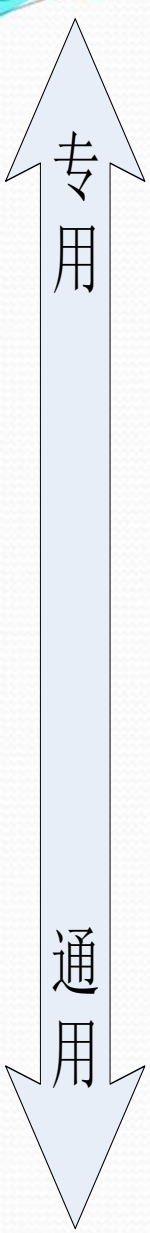
电商金融技术

- 1、电商金融安全简介
- 2、云计算简介
- 3、大模型简介

云计算的定义

云计算是一种商业计算模型。它将计算任务分布在大量计算机构成的资源池上，使各种应用系统能够根据需要获取计算力、存储空间和信息服务。

云计算的类别



将软件作为服务
SaaS (Software as a Service)

如：Salesforce online CRM

将平台作为服务
PaaS (Platform as a Service)

如：Google App Engine
Microsoft Windows Azure

将基础设施作为服务
IaaS (Infrastructure as a Service)

如：Amazon EC2/S3

Google如何实现？

- Google云计算平台技术架构
  - 文件存储，Google Distributed File System，GFS
  - 并行数据处理MapReduce
  - 分布式锁Chubby
  - 结构化数据表BigTable

Google设计GFS的动机

- Google需要一个支持海量存储的文件系统
  - 购置昂贵的分布式文件系统与硬件？

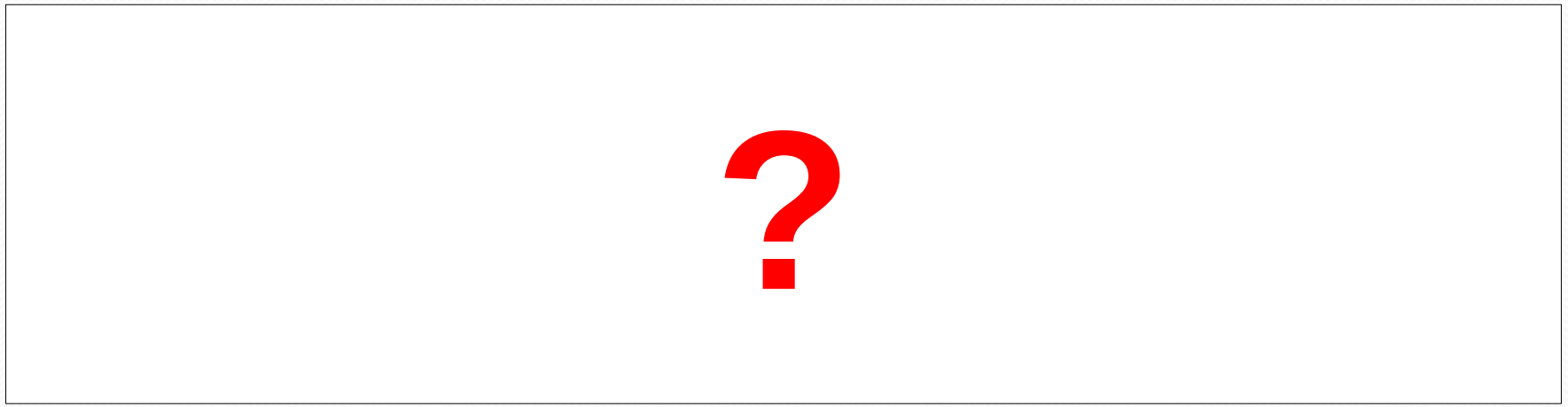
是否可以在一堆廉价且不可靠的硬件上构建可靠的分布式文件系统？

Google文件系统(GFS)

- 将文件划分为若干块（Chunk）存储
  - 每个块固定大小（64M）
- 通过冗余来提高可靠性
  - 每个数据块至少在3个数据块服务器上冗余
- 通过单个master来协调数据访问、元数据存储
  - 结构简单，容易保持元数据一致性
- 无缓存
  - Why?

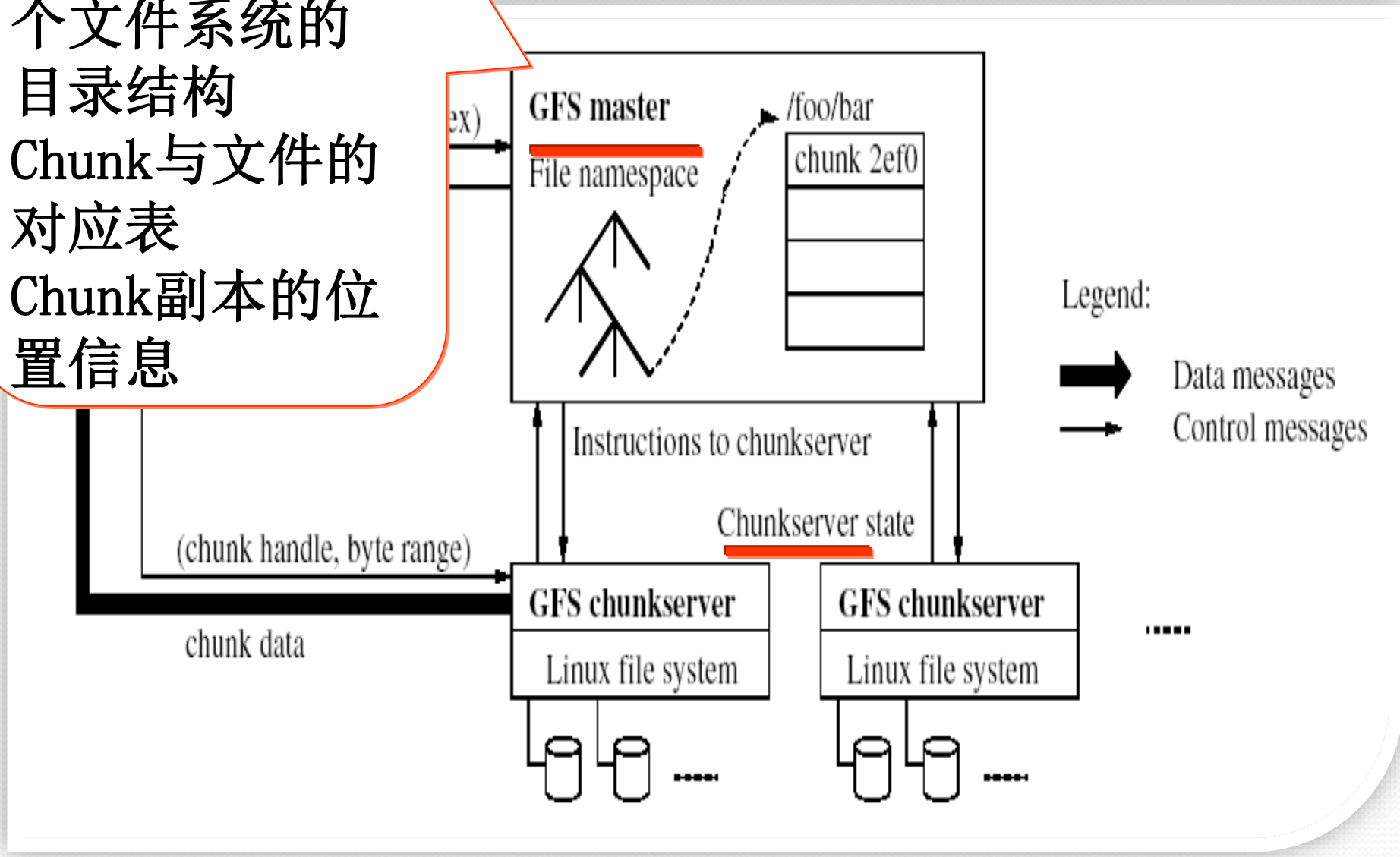
GFS架构的特点

- 不缓存数据
  - GFS的文件操作大部分是流式读写，不存在大量的重复读写，使用Cache对性能提高不大
  - 从可行性看，Cache与实际数据的一致性维护也极其复杂

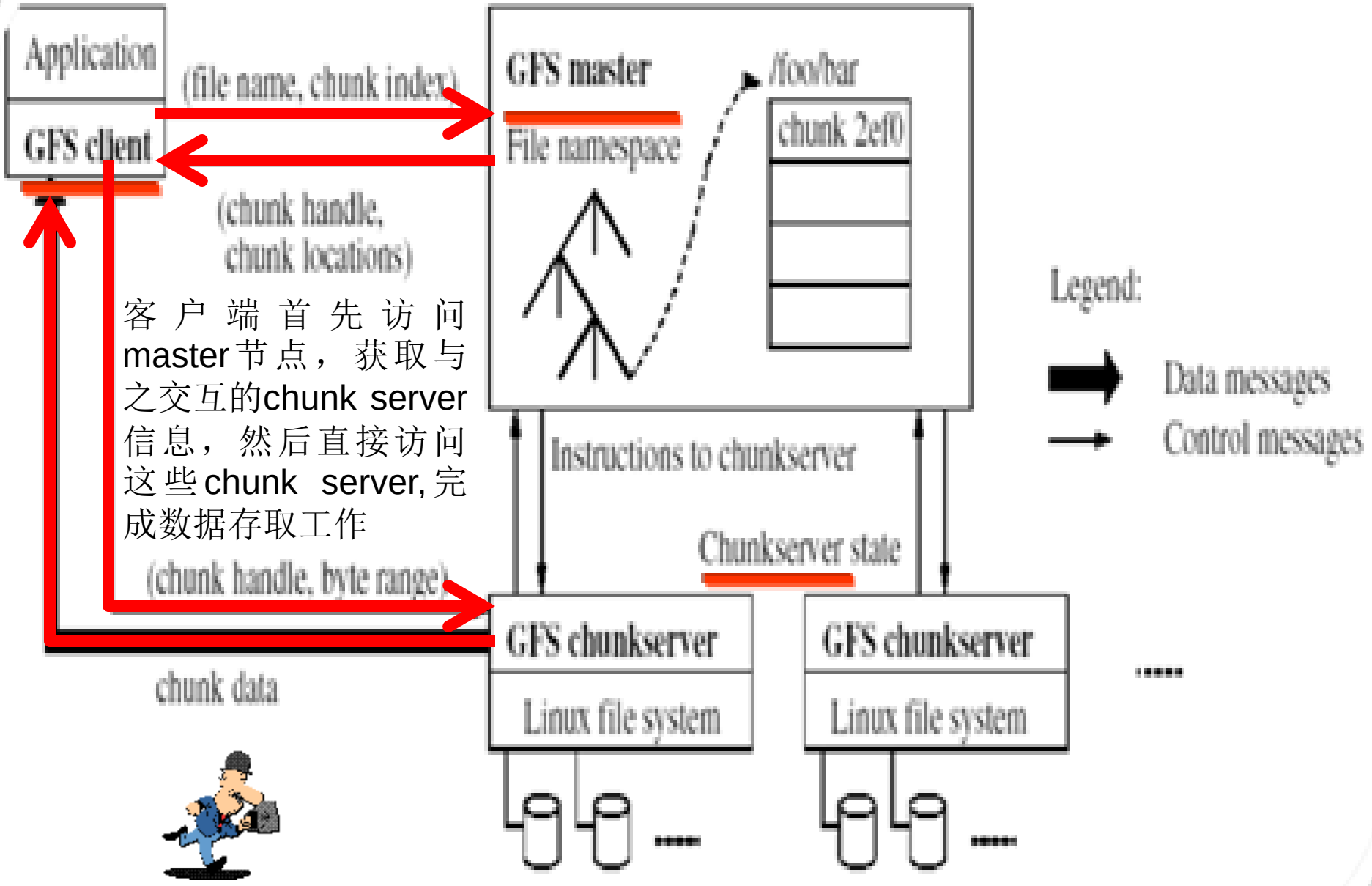


Google文件系统(GFS)

三种元数据
命名空间：整个文件系统的目录结构
Chunk与文件的对应表
Chunk副本的位置信息

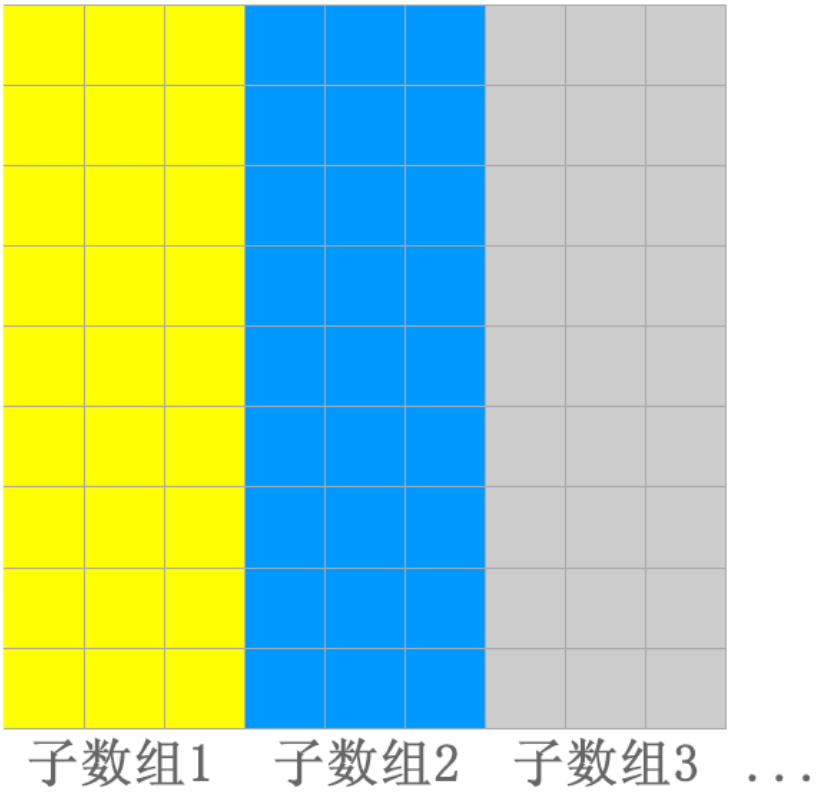


Google文件系统(GFS)



并行计算基础

- 什么样的问题适合并行计算？
  - 如果有大量结构一致的数据要处理，且数据可以分解成相同大小的部分， 那我们就可以设法使这道处理变成并行

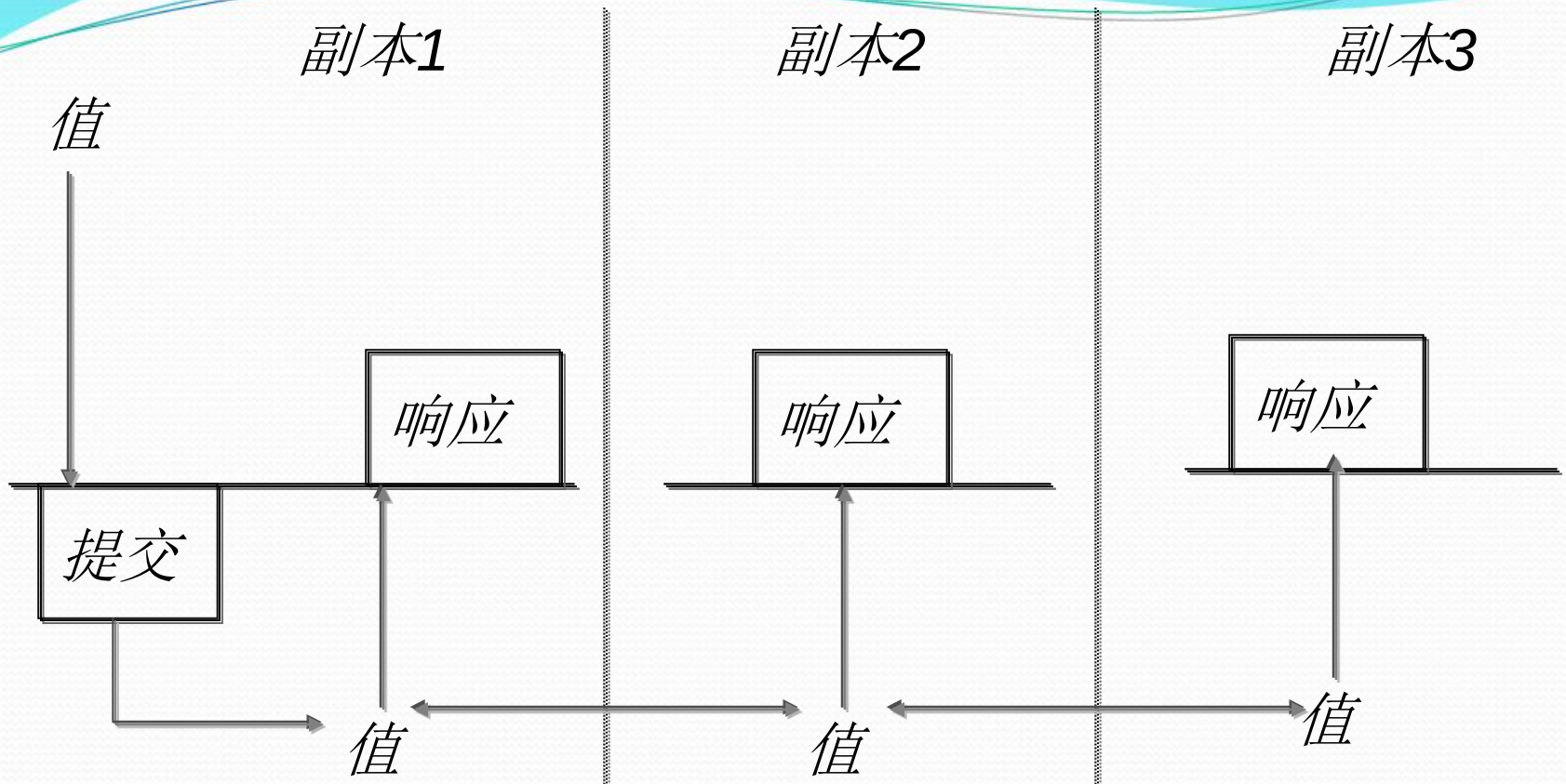


MapReduce

- 一个软件架构，是一种处理海量数据的并行编程模式
- 用于大规模数据集（通常大于1TB）的并行运算
- MapReduce实现了Map和Reduce两个功能
  - Map把一个函数应用于集合中的所有成员，然后返回一个基于这个处理的结果集
  - Reduce对结果集进行分类和归纳
  - Map()和 Reduce() 两个函数可能会并行运行，即使不是在同一的系统的同一时刻

Chubby是什么？

- 主要用于解决分布式一致性问题
  - 在一个分布式系统中，有一组的Process，它们需要确定一个Value。于是每个Process都提出了一个Value，一致性就是指只有其中的一个Value能够被选中作为最后确定的值，并且当这个值被选出来以后，所有的Process都需要被通知到
- 粗粒度的分布式锁服务
  - Chubby是Google为解决分布式一致性问题而设计的提供粗粒度锁服务的**文件系统**



Paxos
构架

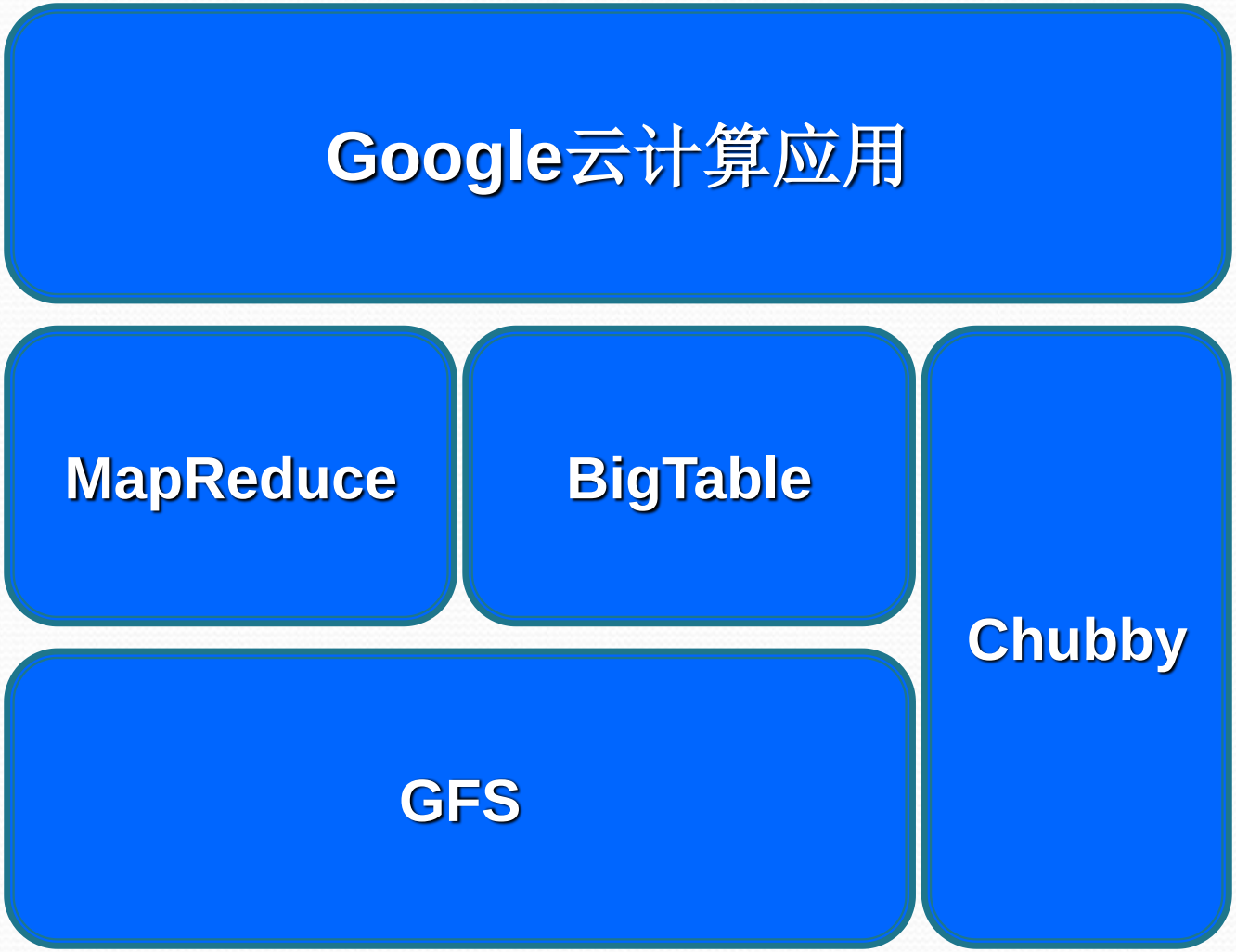
1、选择一个副本成为协调者
2、协调者从客户提交的值中选择一个，然后通过accept的消息广播给所有的副本，接受/拒绝
3、一旦协调者收到大多数副本的接受信息后，就认为达到了一致性。

BigTable

- 为什么需要设计BigTable？
  - Google需要存储的数据种类繁多
    - 网页，地图数据，邮件……
    - 如何使用统一的方式存储各类数据？
  - 海量的服务请求
    - 如何快速地从海量信息中寻找需要的数据？
- BigTable：基于GFS和Chubby的分布式存储系统
  - 对数据进行结构化存储和管理

Google的需求

- 数据存储可靠性
- 高速数据检索与读取
- 存储海量的记录（若干TB）
- 可以保存记录的多个版本


Google云计算应用
MapReduce
BigTable
Chubby
GFS

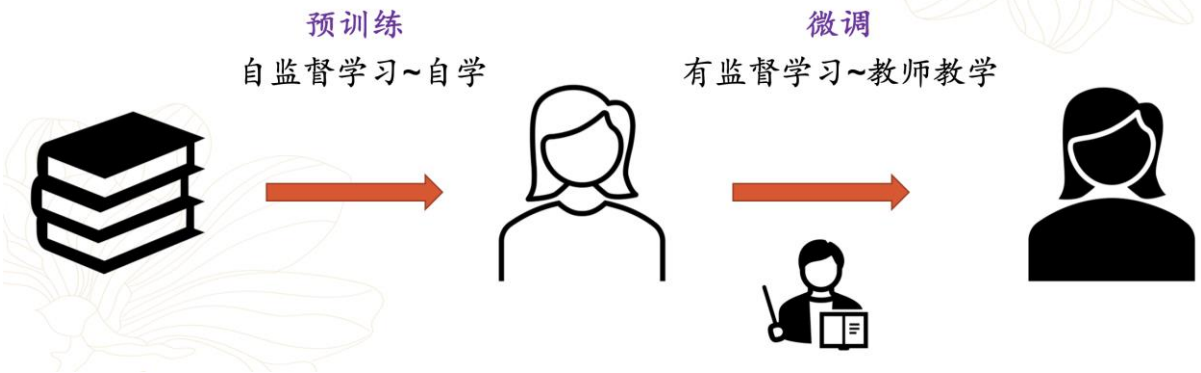
- 1、电商金融安全简介
- 2、云计算简介
- 3、大模型简介

引言

- **背景：**
  - 人工智能领域的巨大飞跃，尤其是大型深度学习模型的兴起，改变了处理信息和数据的方式。
  - 大模型，如GPT-4，代表了这一领域的最新巅峰，具有令人瞩目的上下文理解和学习能力。
- **关键特点：**
  - 超越传统机器学习，大模型在处理复杂任务上表现出色。
  - 通过深度学习，大模型能够从大规模数据中提取抽象模式，为各行业带来新的机遇。
- **为何关注金融领域：**
  - 金融领域对于准确、迅速的信息处理至关重要。
  - 大模型在金融领域的应用有望优化决策、降低风险，并提供更智能的客户服务。

大模型

- 大模型是指参数规模庞大、层次深度的深度学习模型。典型的例子包括GPT-3、BERT等。
- 这些模型利用数以亿计的参数，通过训练数据自动学习任务，无需人为规定特定规则。



提示语的设计

1. **零样本提示（Zero shot）**

角色：希望模型扮演的角色
指令：指定您希望语言模型执行的任务或指令。
上下文：包含相关信息或额外上下文，以帮助语言模型更好地响应。
输入数据：您输入的内容或问题。
输出指示：指定您需要的输出类型或格式。

2. **少样本提示（Few shot）**

通过提供少量的样本就可以让gpt回答的更加精确。

3. **思维链（Chain of thought）**

通过让大模型解释其推理过程，从而实现更加精准的答案。

在金融领域中的应用

AI+金融

谢谢大家！

华东理工大学计算机系
翟洁