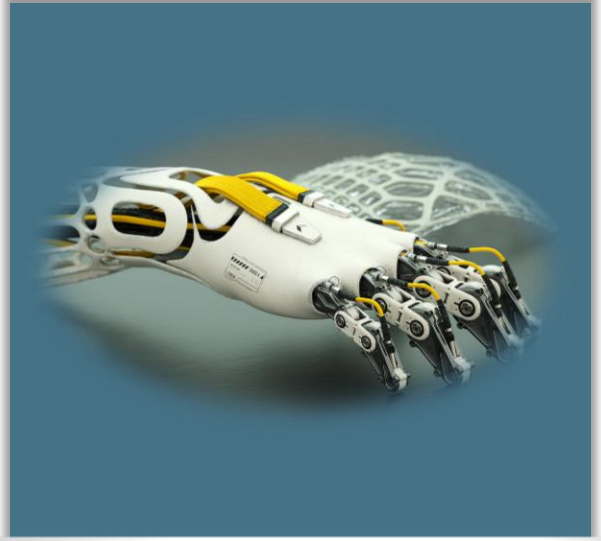
智能感知与物联网

授课人：王闻博

Email: wenbo_wang@kust.edu.cn

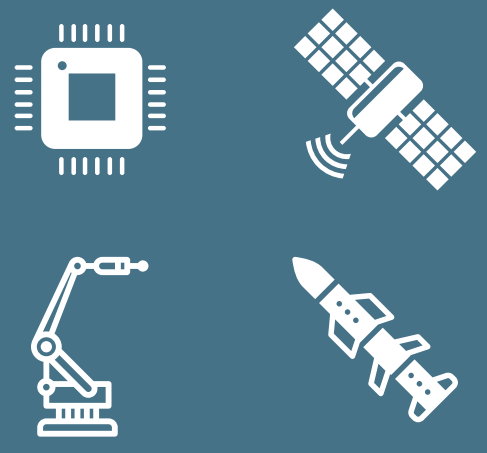
昆明理工大学 机电工程学院

2026年6月02日-6月09日



第七章

面向工业物联网的网络-信息安全

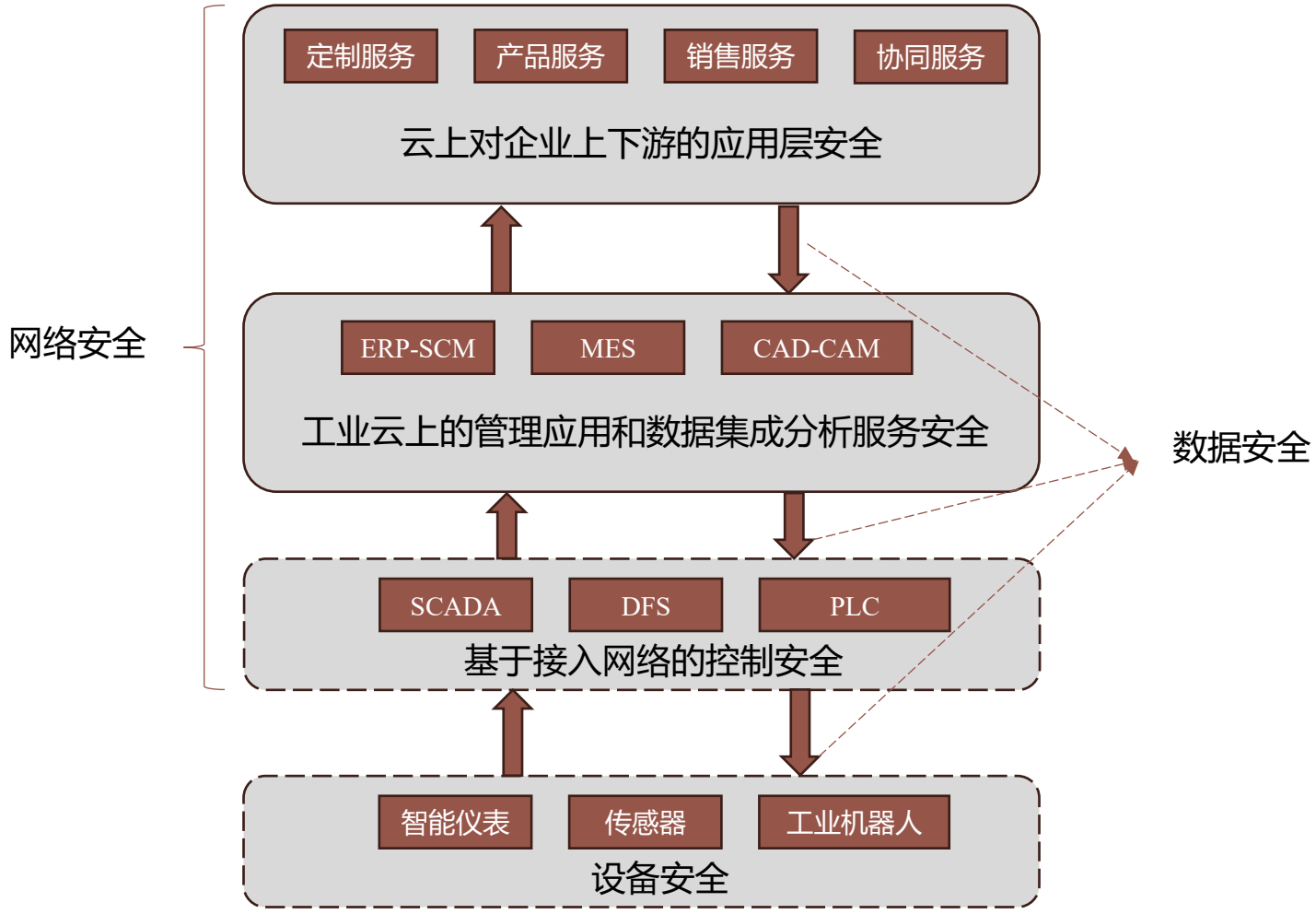


7.1 基于传统加密技术的物联网数据安全

7.2 基于区块链的物联网自组织平台

工业物联网系统的安全层级

- “由底至顶”：
  - **设备安全**：智能制造设备的软硬件安全；
  - **控制安全**：控制协议、平台（Supervisory Control And Data Acquisition，SCADA）、软件等；
  - **网络安全**：工厂内有线/无线网络安全，以及工厂与上下游在公共网络互联的安全；
  - **应用安全**：云端、客户端应用软件的安全；
  - **数据安全**：工厂内的生产管理数据、生产操作数据以及企业留存的外部数据（如用户数据）的安全。



工业生产中物联网系统安全的重要性：一个从网络层面攻击物理设备的实例——Stuxnet

- **Stuxnet（震网病毒）简介**
  - Stuxnet融合了蠕虫、病毒、木马、Rootkit 等多种技术，由以色列和美国的信息战机构研发（世界上首个实战化的网络武器），初始目的是用于破坏伊朗的核设施（铀浓缩泵和离心机），攻击大约发生在2010年。
- **Stuxnet攻击步骤**
  - **初步感染（应用层）**：病毒通过在核工厂内网的一台Windows主机中插入U盘完成第一次感染从而在内网传播。它同时使用了4个0day漏洞进行传播和权限提升，利用用户模式和内核模式的代码进行rootkit攻击，并安装了从Realtek（中国台湾）和JMicron 公司（中国台湾）盗取的经过正确签名和认证的设备驱动，用来规避反病毒软件的扫描。
  - **传播（网络层）**：一旦安装，Stuxnet在Windows系统内搜索典型的西门子SCADA（**监控与数据采集系统**）控制器服务，WinCC/PCS 7 SCADA（也称Step-7）。一旦发现对应服务，它会尝试使用控制与命令服务通过一个错误网址（www.mypremierfutbol.com）更新版本。
  - **攻击（控制层、数据层）**：Stuxnet会在文件系统重搜索名为s7otbdx.ddl的库文件，它是Windows与PLC间通信用库文件。Step-7服务中存在一个硬编码的密码数据库，已经被攻击者在另一个0day攻击中破解。Stuxnet将自身注入到WinCC和s7otbdx.ddl之间进行中间人攻击：拦截并篡改从工程师站下载到 PLC 的逻辑程序，同时又能阻断 PLC 返回的真实状态信号。病毒开始记录离心机的正常运转状态。
  - **破坏（设备层）**：Stuxnet向SCADA上载预先录制的离心机状态，但指令PLC将离心机的转速逐渐（以15到50分钟的周期）设置在高于安全范围的水平，最终导致伊朗核工厂大约1000台离心机的转子被破坏。

网络安全中的常见攻击种类

- **中间人攻击（Man-In-The-Middle）**
  - 攻击者置于相互信任的双方的通信流之间，通过劫持会话（如DNS劫持、Wi-Fi仿冒等），嗅探和监听网络通信，然后伪造一个与被攻击双方之间的通信链路，秘密地截获、篡改或监听它们的通信内容。
- **Rootkit**
  - 通过加载特殊的驱动和修改系统（如利用缓存区溢出）内核来达到隐藏自身、操纵系统和收集数据等目的。典型如智能手机和游戏机破解工具。
- **0day漏洞**
  - 利用尚未被公众所知、且没有官方补丁的漏洞进行攻击，因为系统管理员和安全软件还未意识到这些漏洞的存在，从而无法及时防御。
- **字典攻击**
  - 攻击者使用一个包含大量可能密码的列表（即“字典”），然后使用自动化工具尝试用字典中的每一个密码去登录目标账户，直到找到正确的密码为止。它是穷举法或暴力破解法的一种变体。
- **DDoS攻击（Distributed Denial-of-Service，分布式拒绝服务）**
  - 通过大量的来自僵尸机器（BotNet）的请求流量来拥塞目标服务器、服务或网络，从而使其无法正常提供服务。

网络安全中的常见攻击种类（续）

- **返回库函数（Return-to-libc）攻击**
  - 返回库函数攻击的原理是利用程序中的漏洞，通过缓冲区溢出等手段，改写程序函数调用的返回地址，使其指向库函数（如libc库中的system()函数）的地址。同时，攻击者还会重新设定库函数执行时的参数，以便在函数返回时执行攻击者期望的代码序列，而不是返回到程序的主函数。
- **SQL注入**
  - 攻击者通过在应用程序的输入字段中插入或“注入”恶意的SQL代码，从而在未经授权的情况下操作数据库。
- **社会工程（Social Engineering）**
  - 通过与他人的合法交流，影响其心理，使其做出某些动作或透露机密信息。这通常被视为欺诈行为，用于收集信息、行骗和入侵计算机系统。常见方式如钓鱼邮件攻击、社交媒体欺诈等。
- **跨站脚本攻击（XSS）**
  - 在目标网站上注入恶意脚本。当用户浏览该网站时，恶意脚本会在用户的浏览器中执行，从而窃取用户的信息或者进行其他恶意操作。
- **边信道（Side-Channel）攻击**
  - 攻击者通过分析加密软件或硬件在运行时产生的各种泄漏信息，如能量损耗、电磁辐射、运行时间等，来获取密文信息或密钥的部分信息。

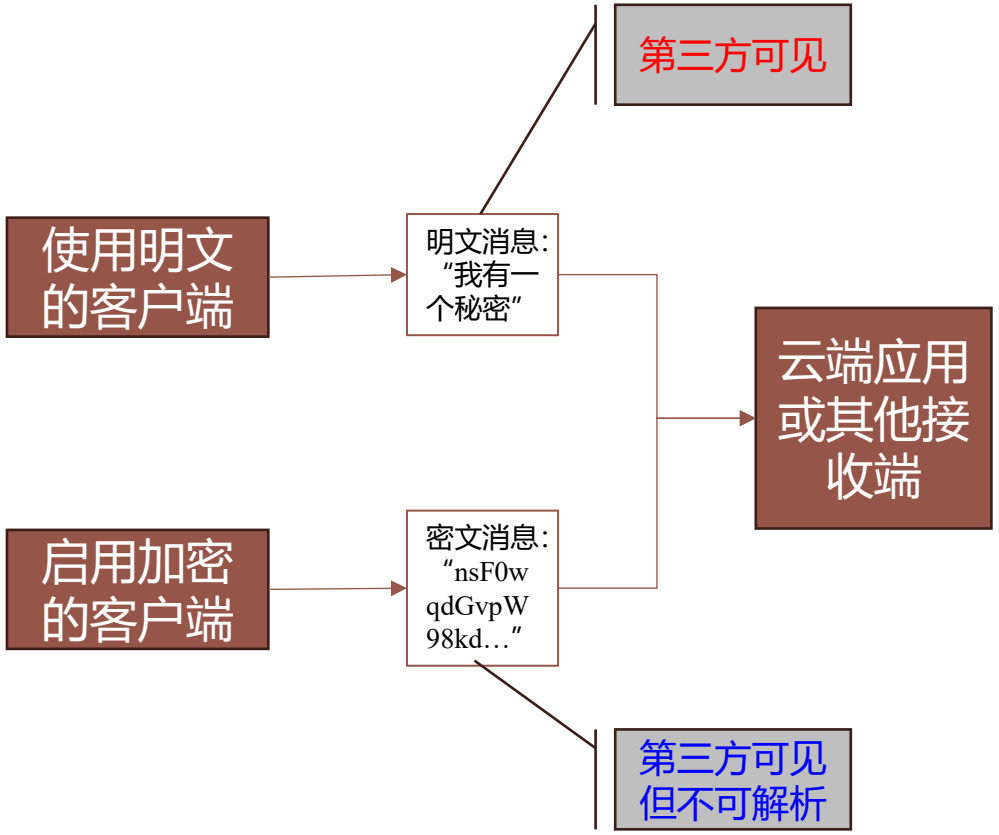
网络安全的通用解决手段

- **物理安全**
  - 通过对设备、通信线路及网络的物理保护和隔离，如完全断开和公共网络的物理连接的基础上，实现安全受控的内网信息共享。
- **用户身份认证**
  - 身份认证是验证用户提供的访问凭证是否有效和合法的过程。如通过动态密码、数字证书、设备指纹识别（基于MAC地址、系统类型等的设备特征识别）等机制，核验连接请求，对非法连接进行阻断。
- **防火墙**
  - 通过在内网和不安全外网之间设置访问控制，对流量进行过滤、清洗再转发到接收端（如服务器），从而达到防止SQL注入、跨站脚本攻击、木马攻击等常见攻击形式。
- **入侵检测系统（Intrusion Detection System，IDS）**
  - 通过对网络传输进行即时监视，对网络数据包流量进行监测和异常行为分析来发现潜在入侵行为。
- **入侵防御系统（Intrusion Prevention System，IPS）**
  - 通过在线分析，如对威胁的统计或特征检测，阻止来自网络的入侵行为。

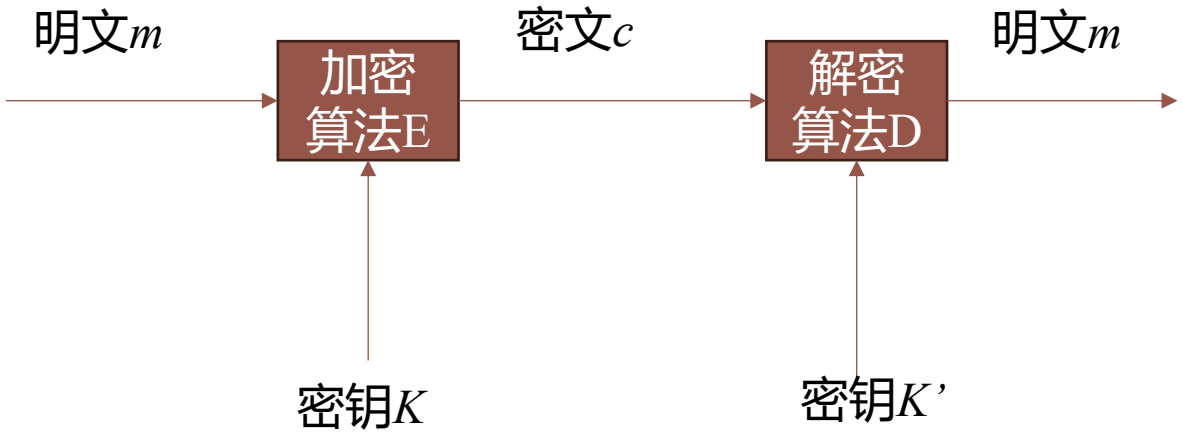
网络安全的通用解决手段（续）

- **安全路由器**
  - 是防火墙的补充，通过访问控制列表等技术控制网络信息流。
- **安全服务器**
  - 针对局域网内部信息存储、传输的安全保密问题设置。实现对局域网资源的管理和控制，以及所有内网相关安全事件的审计和跟踪
- **安全数据库**
  - 确保数据库的完整性、可靠性、有效性、机密性、可审计性，以及存取控制和用户身份识别等。
- **安全硬件和安全操作系统，主要技术有**
  - 可信执行环境（Trusted Execution Environment，TEE）：处理器中的一个安全区域，确保在此区域内的代码和数据受到保护，主要服务于安全启动、私钥处理等。
  - 可信根（Root of Trust，RoT）：在冷启动设上从不可变的、可信内存（如ROM）开始执行。
  - 地址空间配置随机化：由大多数主流操作系统支持，通过随机放置进程关键数据区域的地址空间，防止攻击者能可靠地跳转到内存的特定位置来利用函数。这是一种针对缓冲区溢出的安全保护技术，通过对堆、栈、共享库映射等线性区布局的随机化，增加攻击者预测目的地址的难度，从而防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的。

信息安全的起点：加密算法



- **一般加密模型：**



- **加密和解密变换的关系**

$$c = E_K(m),$$

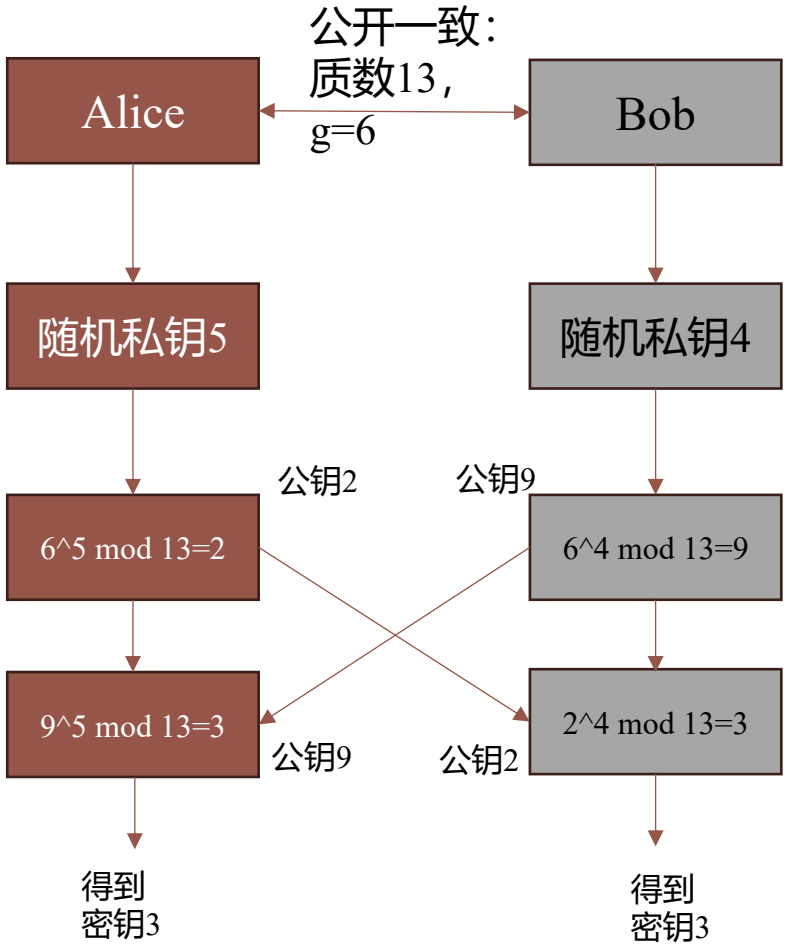
$$m = D_{K'}(c) = D_{K'}(E_K(m))$$

加密算法简介

- **对称加密**
  - 单密钥加密，即加密解密的密钥是相同的。
  - 步骤：
    - （1）数据发送方将明文（原始数据）经过特殊加密算法处理后，使其变成复杂的加密密文发送出去。
    - （2）接收方收到密文后，需要使用同一加密密钥及相同算法的逆算法对密文进行解密，才能使其恢复成可读明文。
  - 示例：高级加密算法（AES）或ISO/IEC 18033-3标准，其使用128、192或256位的固定密钥长度，采用分组等长处理模式（每组16字节），每个明文组加密后产生一个等长的密文组（不足16字节则需要填充）。
  - 缺陷：密钥的管理和分发困难。在数据传送前，发送方和接收方必须商定好密钥，并且双方都要保存好密钥。如果一方的密钥被泄露，那么加密信息就会变得不安全。

密钥交换协议

- **对称加密密钥交换：Diffie-Hellman交换过程**
  - **选择参数：选择一个大质数p和一个原根g（质数生成器），这两个参数可以公开。**
  - **生成私钥：通信双方Alice和Bob各自选择一个秘密的随机数a（Alice）和b（Bob）作为私钥，不公开。**
  - **计算公钥：Alice计算 $A = g^a \mod p$，Bob计算 $B = g^b \mod p$，然后双方交换公钥。**
  - **计算共享密钥：Alice使用她的私钥a和Bob的公钥B来计算 $s = B^a \mod p$；同时，Bob使用他的私钥b和Alice的公钥A来计算 $s = A^b \mod p$。由于数学上的性质，双方计算出的s将会是相同的值，这个值就可以作为双方的共享密钥。**



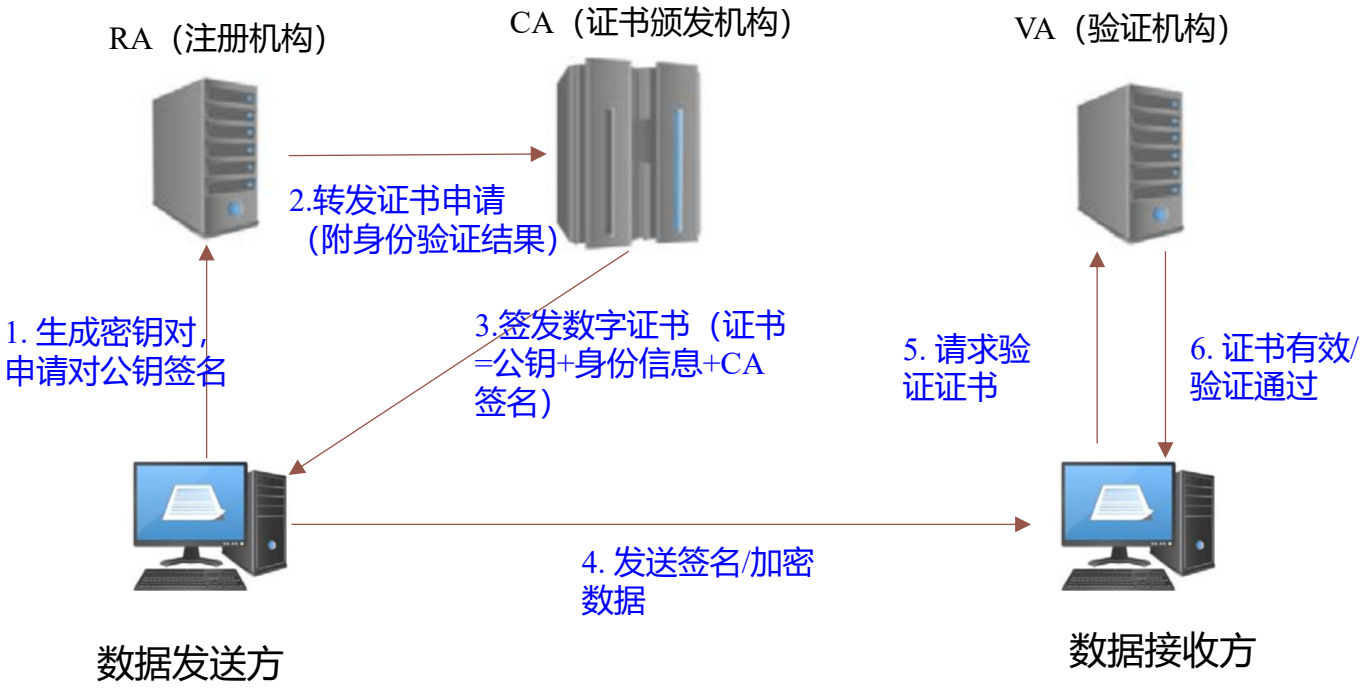
加密算法简介（续）

- **非对称加密（或称公钥加密）**
  - **加密密钥**（*K*）和**解密密钥**（*K'*）不同，形成一个密钥对（ *K* ， *K'* ）。
  - **特点**：密钥在加-解密过程中可以互换，即使用这个密钥对的时候，如果用其中一个密钥*K*加密一段数据，必须用另一个密钥*K'*解密，反之也成立（**密钥对中的一种密钥加密的数据必定能使用另一种密钥解密**）。
  - **使用**：保持一对密钥中的“**私钥**”由所有者持有，不可公布；而“**公钥**”由密钥对持有者公布给他人。操作中，公钥用来给数据加密，接收方收到的密文只能使用私钥解密。
  - **摘要**：对需要传输的不定长文本，做一次**HASH**计算得到定长镜像值（即摘要Digest），一般采用SHA（如SHA-1，SHA-256）算法获得。
  - **签名**：使用私钥对需要传输的**文本的摘要**进行加密，得到的密文即被称为该次传输过程的**签名**。接收端收到**传输文本**和**签名**后，再通过公钥对签名进行解密，得到了**文本的摘要**，然后使用与发送方同样的**HASH**算法计算**传输文本**摘要值，再与解密得到的摘要做对比。如果二者一致，则说明文本没有被篡改过。

非对称加密中公钥的管理与分发：公钥基础设施

- **公钥基础设施（Public Key Infrastructure）体系**
  - **由受信第三方管理创建和分发数字证书。**
  - **用于管理HTTPS中的SSL（Secure Sockets Layer）和TLS（Transport Layer Security）证书。**
- **组成部分**
  - **密钥对生成器：**在网站/服务器本地，用于生成公钥（公开发布）和私钥（严密保管）对。
  - **注册机构（RA）：**负责处理用户证书申请（包含公钥和身份信息）、身份审核等任务，审核内容包括核实身份信息（如域名）的声称者，确实是该身份信息的合法拥有者。
  - **证书颁发机构（CA）：**负责签发和管理数字证书，是PKI的核心部分。用户通过注册机构（RA）提交申请，RA 审核通过后，由 CA 使用其根私钥为用户的公钥进行数字签名以生成证书。因此，RA 可看作是用户与 CA 之间的中介，而 CA 则是整个信任体系的后端权威。
  - **证书存储库：用于存储和管理数字证书。**

PKI证书管理流程



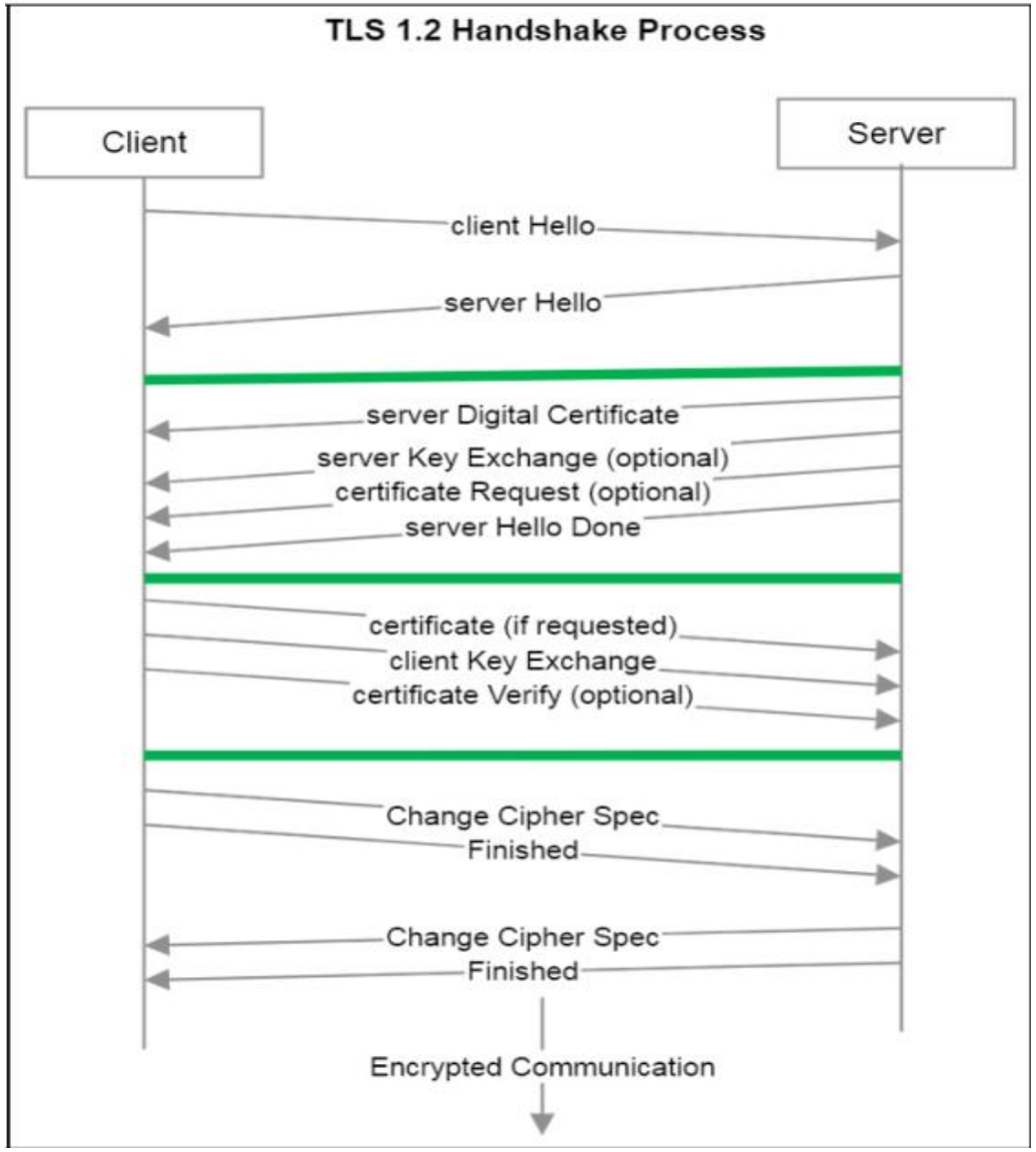
- **密钥的生成和证书颁发过程**
  - **生成私钥：使用OpenSSL生成一个私钥：**
    - **Bash代码：** `openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048`
  - **生成公钥：**
    - **Bash代码：** `openssl rsa -pubout -in private_key.pem -out public_key.pem`,
  - **创建证书签名请求：**
    - **Bash代码：** `openssl req -new -key private_key.pem -out csr.csr`
  - **将签名请求（.csr文件）提交给CA**

基于公钥的客户端-服务端会话密钥生成

- 使用TLS（传输层安全，Transport Layer Security）协议
  - TLS由MQTT和CoAP等协议广泛使用。
- 会话密钥（Session Key）生成过程
  - 随机数生成：在TLS握手开始阶段，客户端和服务器都需要生成一个随机数。这些随机数在后续的握手过程中用于生成对称加密算法的密钥。
  - Client Hello：客户端向服务器发送一个Client Hello消息，其中包含支持的TLS协议版本号、加密算法套件列表和一个随机数（Client Random）。
  - Server Hello：服务器收到Client Hello后，选择一个合适的加密套件，并生成另一个随机数（Server Random）。服务器发送Server Hello消息给客户端。
  - 证书交换与验证：服务器发送其数字证书给客户端，客户端验证服务器的证书。
  - 密钥交换：客户端用服务器的公钥加密随机数发给服务器，二者再通过随机数创建会话密钥（如使用预共享密钥创建会话密钥），并在通信期间使用此密钥（注意：这里使用对称加密算法）。

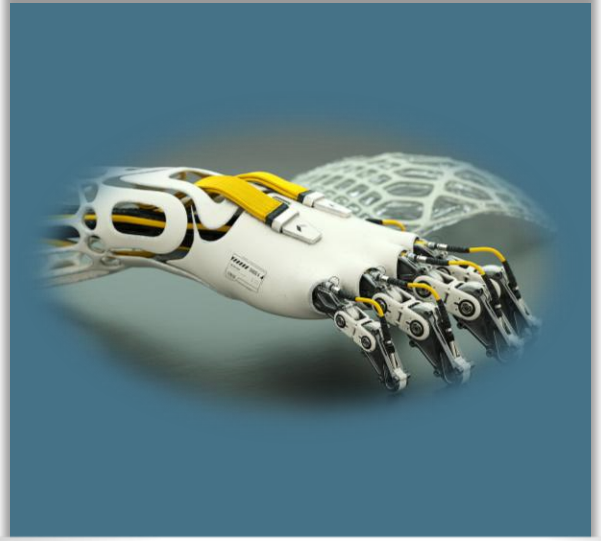
基于公钥的客户端-服务端会话密钥生成（续）

- **注意**
  - **TSL也可使用Diffie-Hellman密钥交换生成数据传输用会话密钥。**
  - **在握手过程中，TSL采用非对称加密；在数据传输过程中，TSL采用对称加密。**



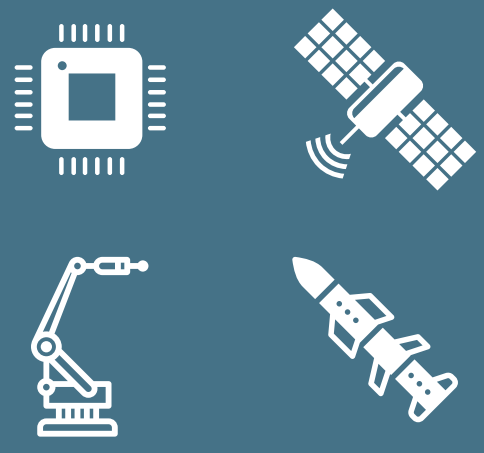
IOT系统中面临的特殊信息安全难题

- **不够安全的通信协议和接口**
  - 接入节点（传感器等）到边缘节点往往采用短距离无线通信，边缘节点到云往往采用消息中间件。这些协议在安全方面考虑不足。另外，分布式架构引入了大量接口，目前相关设计未充分考虑安全性。
- **不安全的系统与组件**
  - 边缘节点有可能分布式分担云端的计算任务，但边缘节点有可能从云端下载不安全的定制操作系统，或是被攻击过的供应链上的第三方软件/固件。因此边缘节点的计算结果在云边协同过程中存在可信性问题。
- **由于算力限制造成的身份、凭证和访问管理不足**
  - 接入设备往往没有足够的存储和计算资源执行认证协议所需的加密操作，为此需要把这些工作委托给边缘节点（网关等）。边缘服务器需要为大规模异构设备提供接入访问控制，目前尚未有统一的解决方案。
- **恶意边缘节点**
  - IoT网络中，恶意节点更容易伪装成合法节点，同时，攻击者更容易利用设备异构、协议和服务提供商的不同发起物理攻击。
- **容易发起的DDoS攻击**
  - 接入设备往往没有足够的存储和计算资源部署完整防御方案，导致攻击者可以轻松入侵这些设备形成僵尸设备（Bot），并利用这些设备发起大流量DDoS攻击。



第七章

面向工业物联网的网络-信息安全



7.1 基于传统加密技术的物联网数据安全

7.2 基于区块链的物联网自组织平台

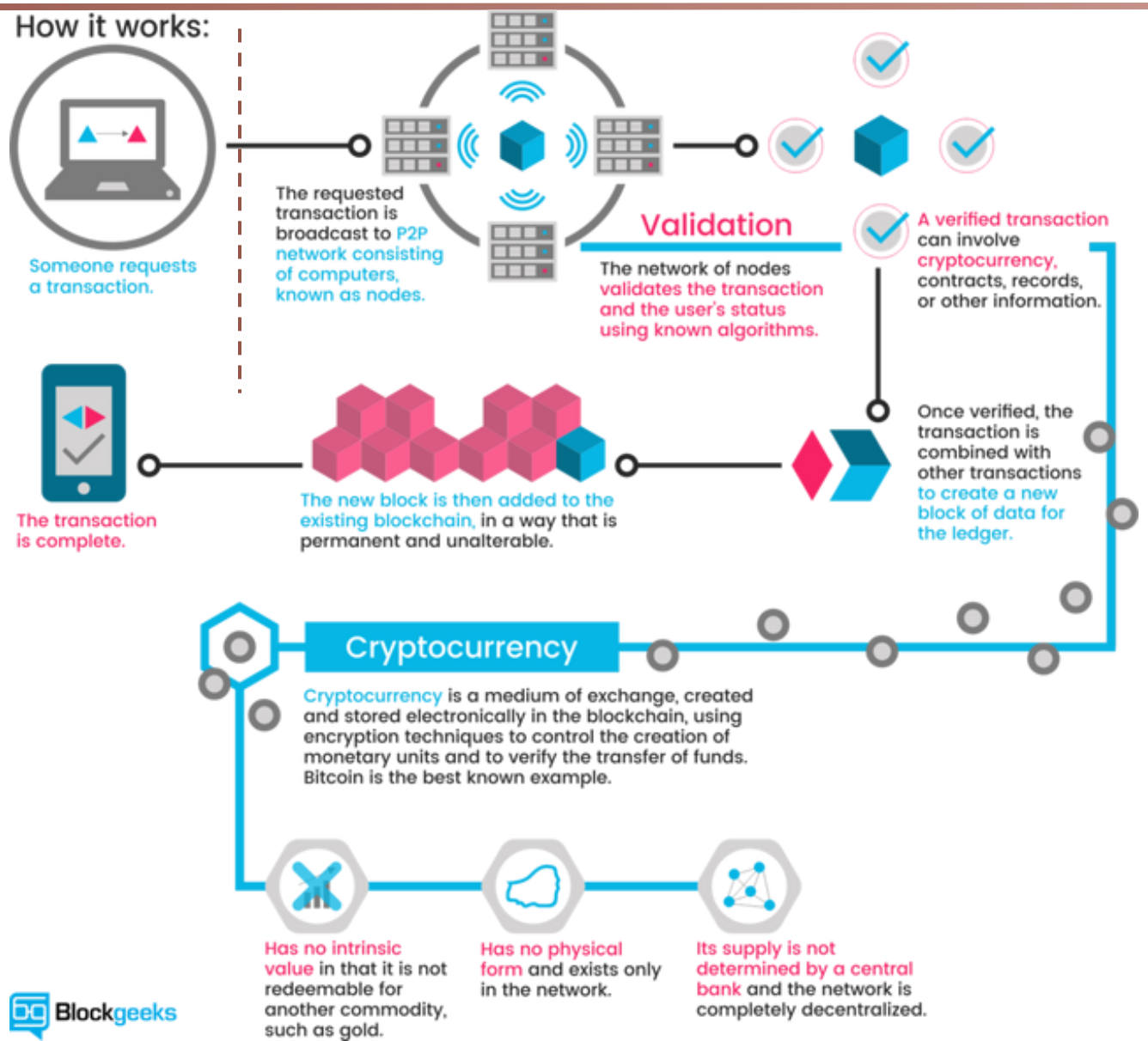
区块链概览

- 什么是区块链?
  - 数据组织层面：区块链提供一种分布式的防篡改数据结构。
  - 网络组织层面：区块链是一种P2P网络。
  - 应用领域：区块链是一种分布式账本系统。
- 区块链为什么能够提供信息安全?
  - 采用多种密码学技术。
  - 使用特殊的异步分布式系统中的一致性达成协议，如Proof-of-Work。
  - 使用“机制设计”迫使自私节点遵循协议。
- 区块链技术的现今发展方向
  - 一致性协议设计。
  - 数据结构设计。
  - 智能合约设计。

区块链工作流程：从比特币（Bitcoin）说起

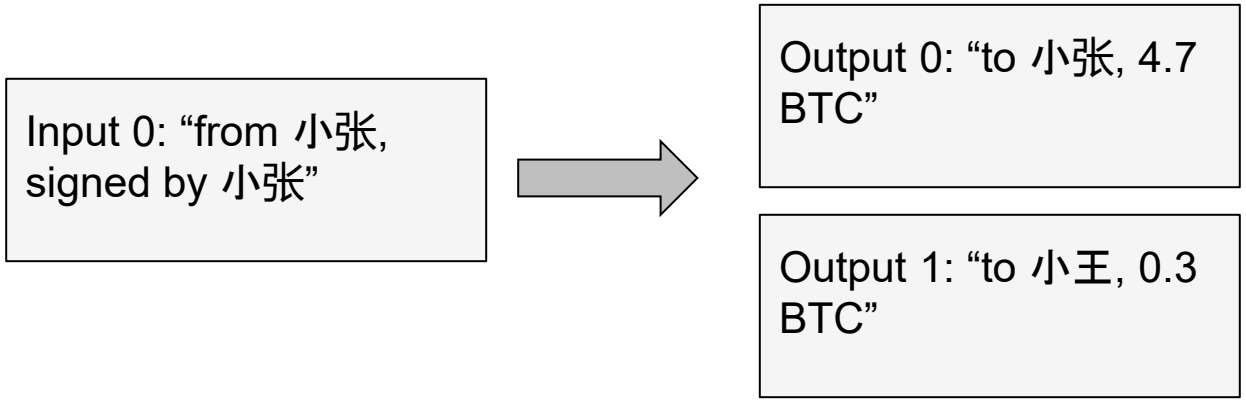
第一步：用户端

第二步：网络端



区块链的Token（代币）交易机制

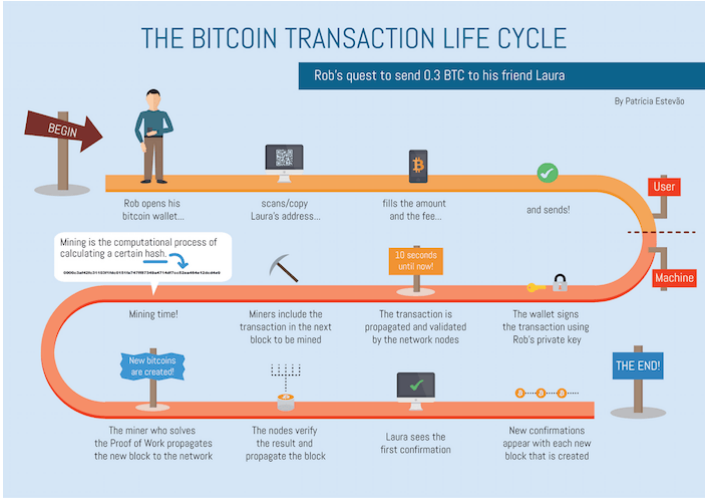
- **小张有5个bitcoins（BTC），他想向小王转账0.3BTC，如何实现?**



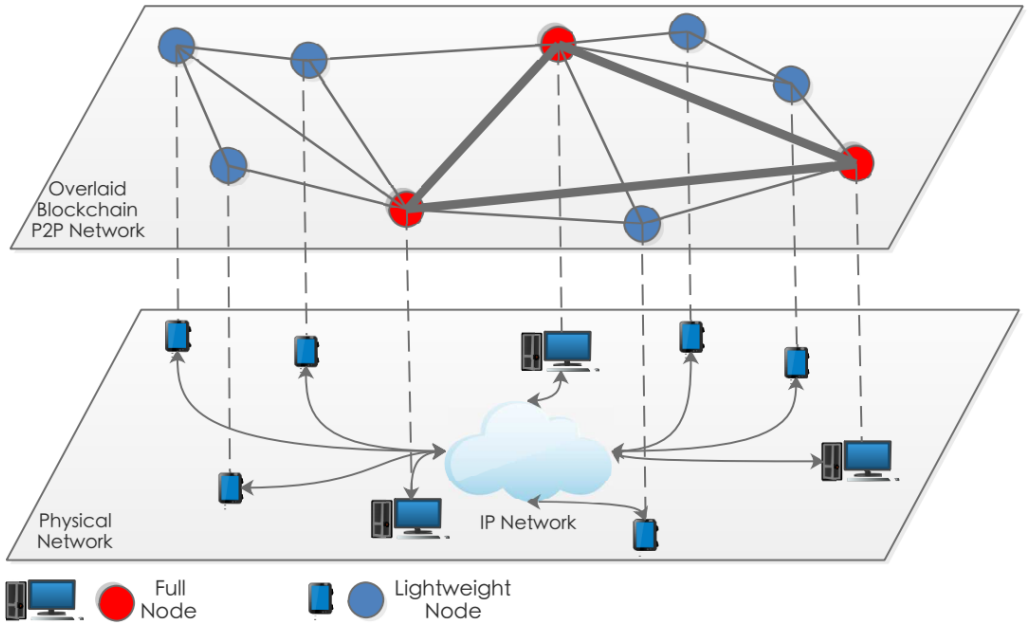
- **依靠区块链技术，我们希望达成：**
  - **Atomicity （原子性）：交易要么确认要么失败。**
  - **Consistency （一致性）：账户由于交易从一种状态转变为另一种状态，中间不会产生状态冲突。**
  - **Isolation （隔离性）：不受其他并发事务的干扰/不对其他并发事务造成干扰)。**
  - **Durability （永久性）：一旦提交，所有账户状态转换产生的影响都是永久的。**

区块链：一个Overlay P2P网络

- **交易确认过程（Transaction）中发生了什么?**
  - **交易数据验证。**
  - **数据由共识节点打包。**
  - **数据由共识节点向网络扩散。**
  - **通过某种一致性达成过程对数据进行全网确认。**

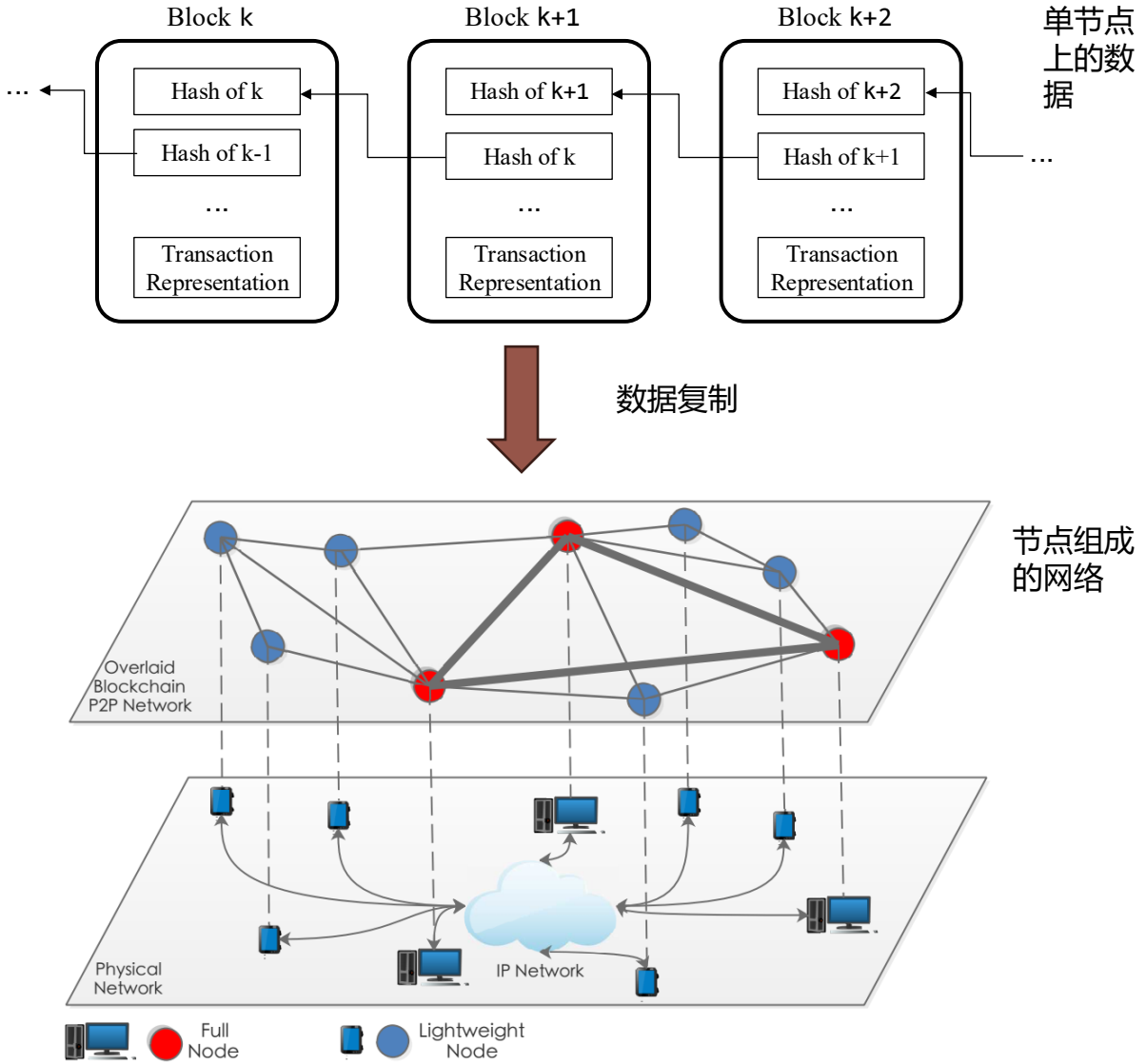


- **交易中的参与方**
  - **轻节点（Light weight nodes）：钱包或交易发起方。**
  - **全节点（Full nodes）：如网络中的共识节点（又名，区块矿工或交易验证节点）。**



区块链的两种定义

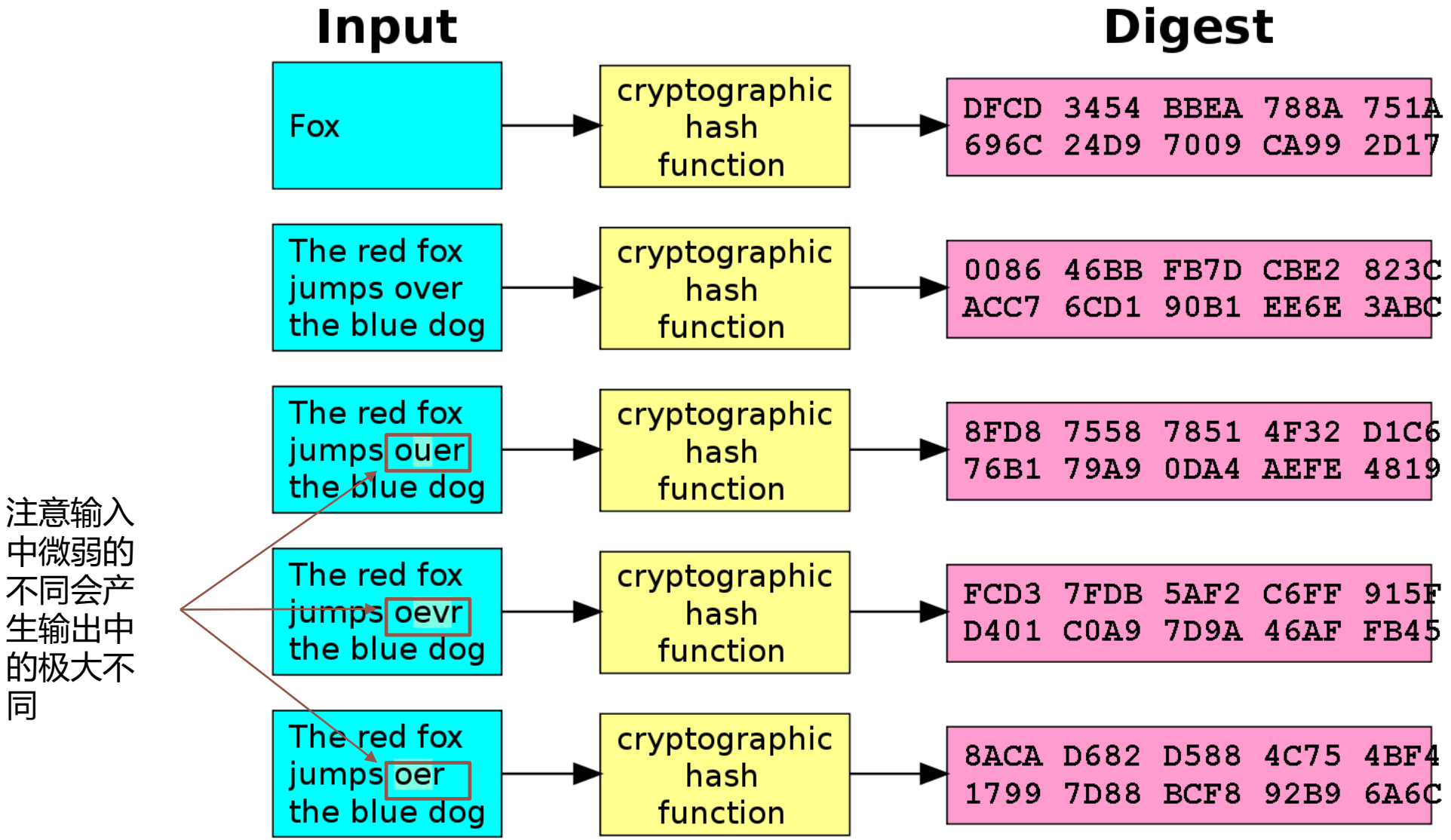
- **作为数据结构的区块链**
  - **是链式的区块数据组织结构，通过加密算法保证不可篡改。**
- **作为节点P2P网络的区块链**
  - **通过数据传输协议、共识协议保证主链数据的全网复制和维护。**



作为数据结构的区块链：关键技术

- **Hash函数：**
  - **一种产生不可预测（随机）结果的确定性单向函数。**
  - **例子：MD5、SHA256、SHA512。**
- **伪身份（Pseudo-identity）**
  - **采用非对称加密算法生成公有和私有密钥。**
  - **采用数字签名确保数据可验证。**
- **交易记录（Transactions）**
  - **代币归属验证机制。**
  - **交易发起-确认机制。**
- **区块的链式结构**
  - **使用Hash指针和Merkle树。**

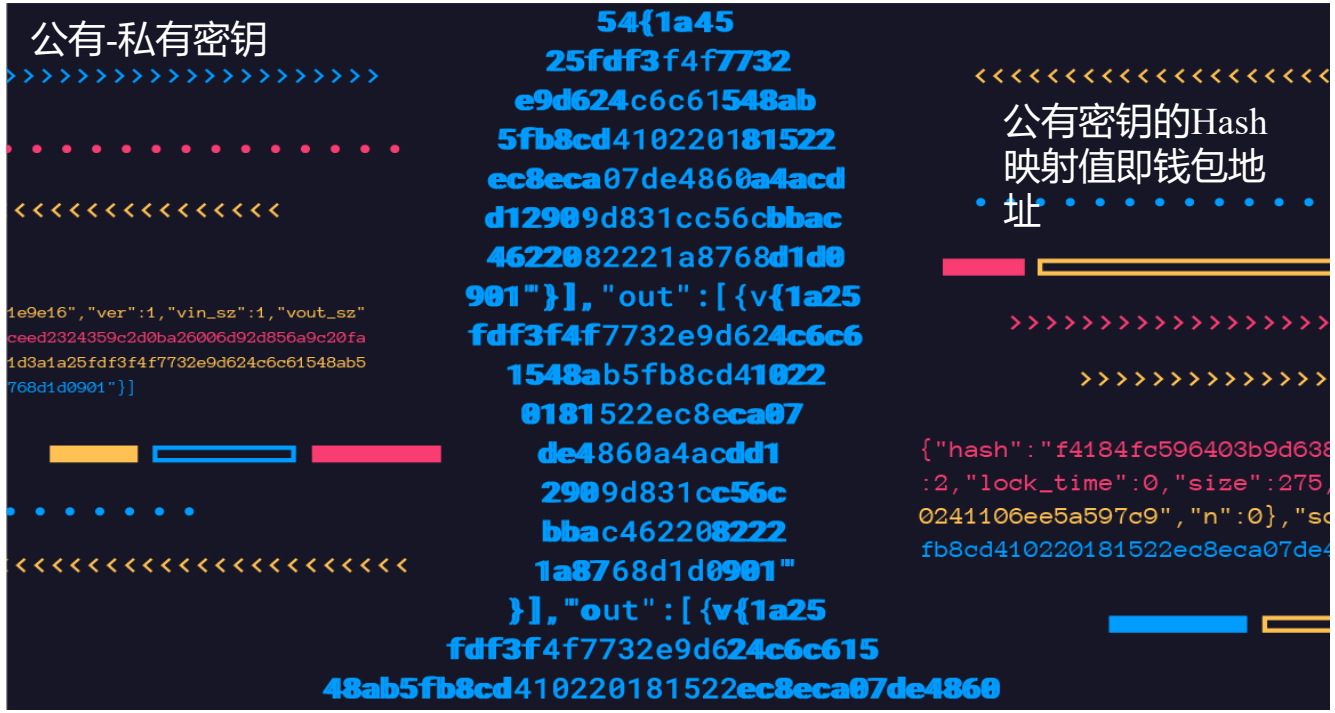
Hash函数示例



注意输入中微弱的不同会产生输出中的极大不同

公有-私有密钥和Pseudo-identity

- **账户和秘钥的关系：**
  - **用户持有的账户由公钥和私钥加密对组成。**
  - **私钥用于签名交易。**
  - **公钥通过椭圆曲线算法由私钥映射生成，一个私钥只能生成一个公钥。**
  - **伪身份：公钥的Hash值被作为账户的地址，一台主机（一个IP地址）可以拥有多个账户地址。**
  - **账户（以以太坊为例）：用来存储对应地址的余额（balance），发起交易和创建的智能合约数量（nounce）等。**



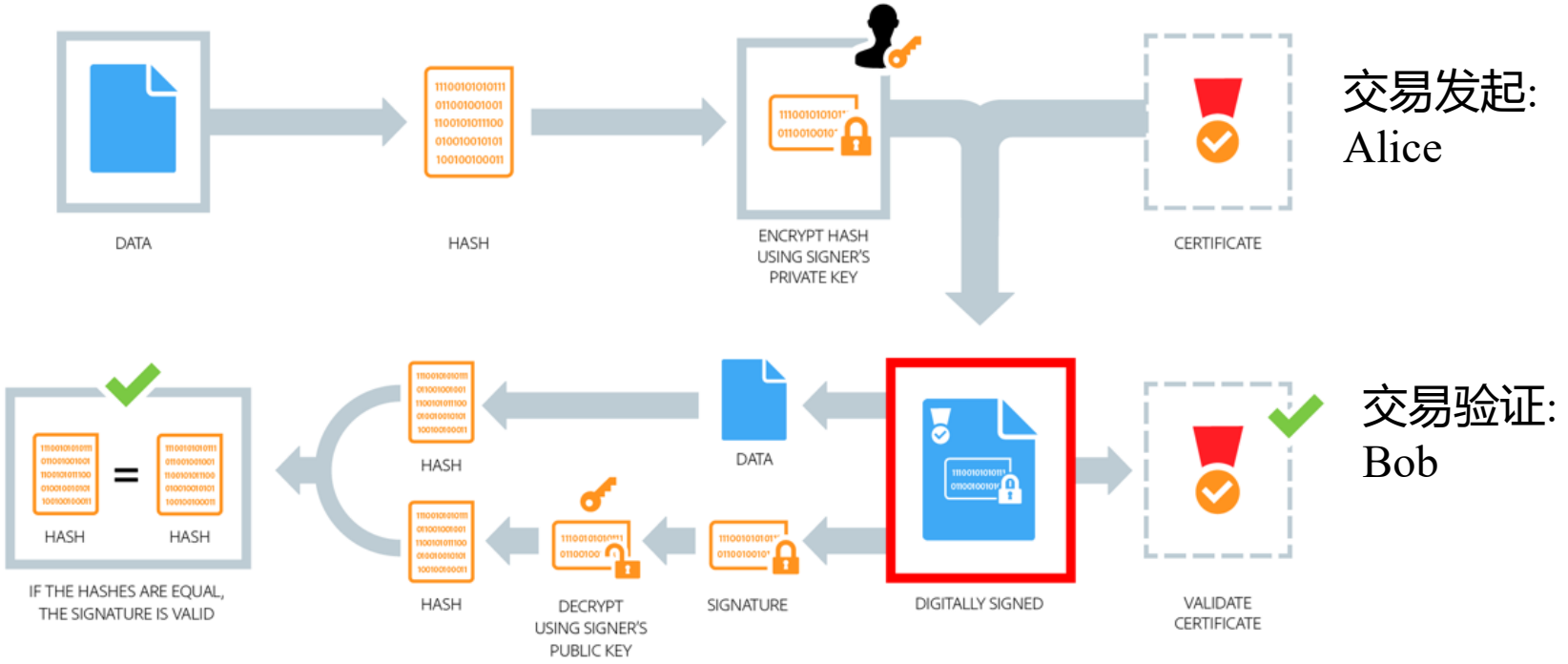
- **秘钥生成**
  - **（以太坊）使用密码和keyfile共同生产私钥：利用Geth（以太坊客户端的账户管理和签名工具）通过指定密码生成keyfile和私钥，典型JavaScript代码如**

```
web3.personal.newAccount("123456").
```

私钥-公钥对和区块链节点地址的的生成过程：以以太坊为例

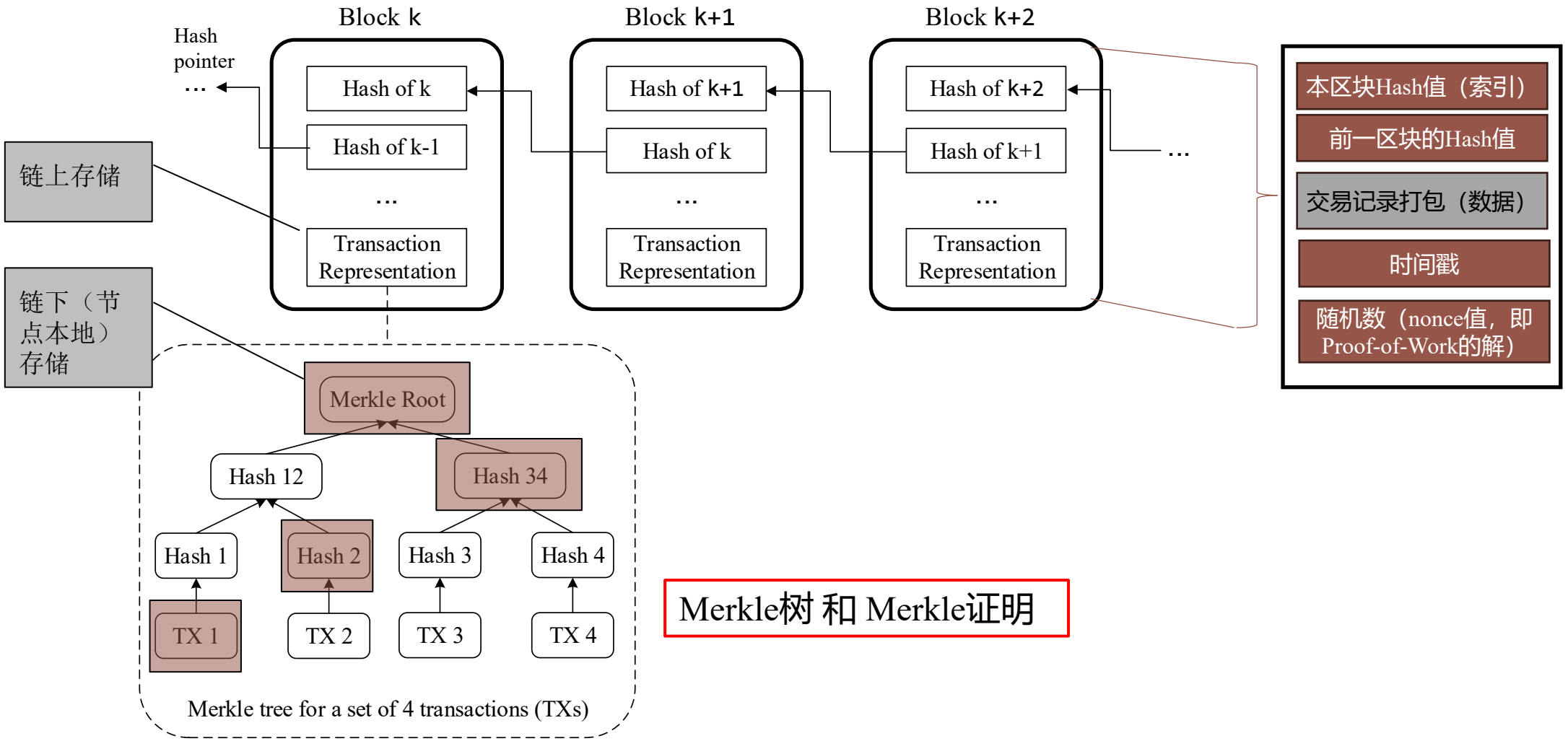
密钥的使用：数字签名

Public Key	Private Key
信息加密	信息解密
签名验证{真或假}	信息签名
通过Hash函数生成账户地址	N/A



Source: https://blogs.adobe.com/adobeingovernment/how-digital-signatures-work-under-the-hood/

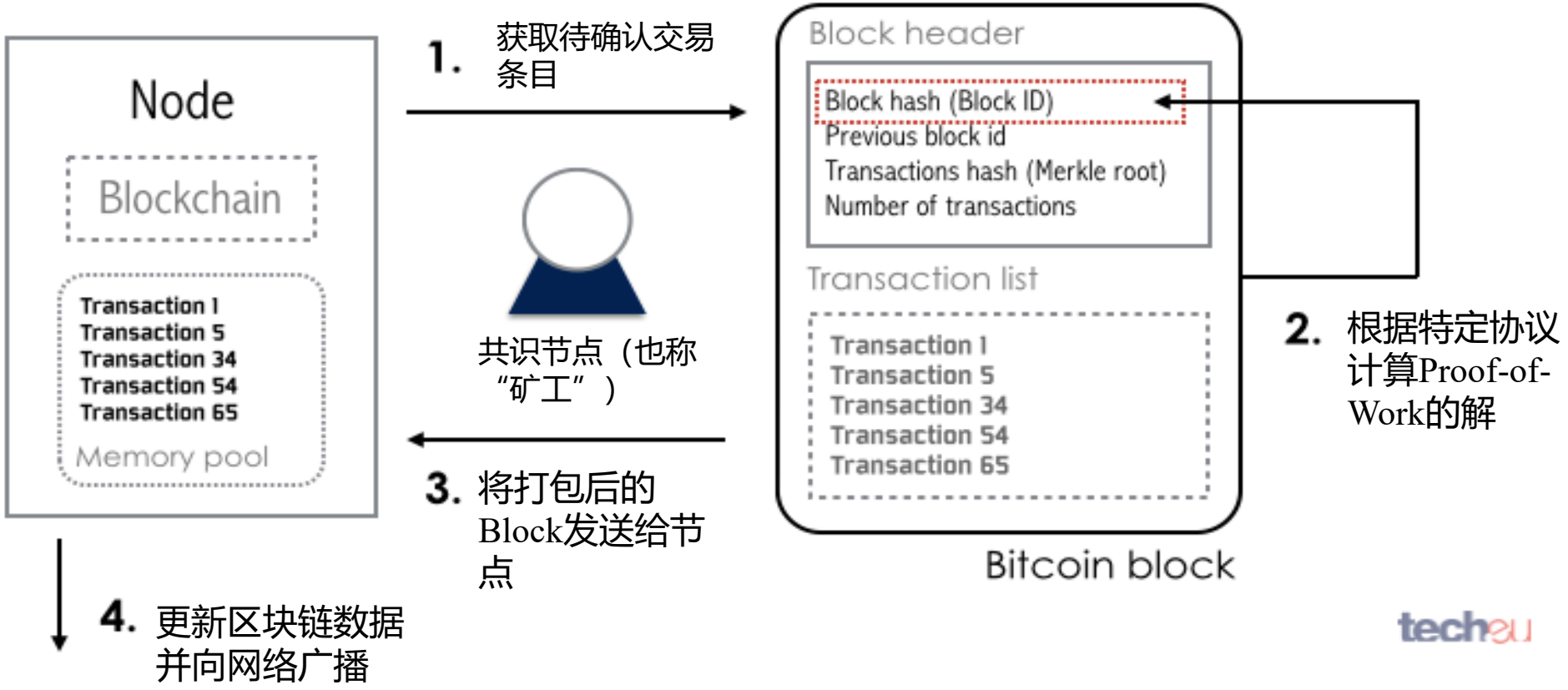
区块链在单一节点上的数据组织：Merkle树和Hash链表



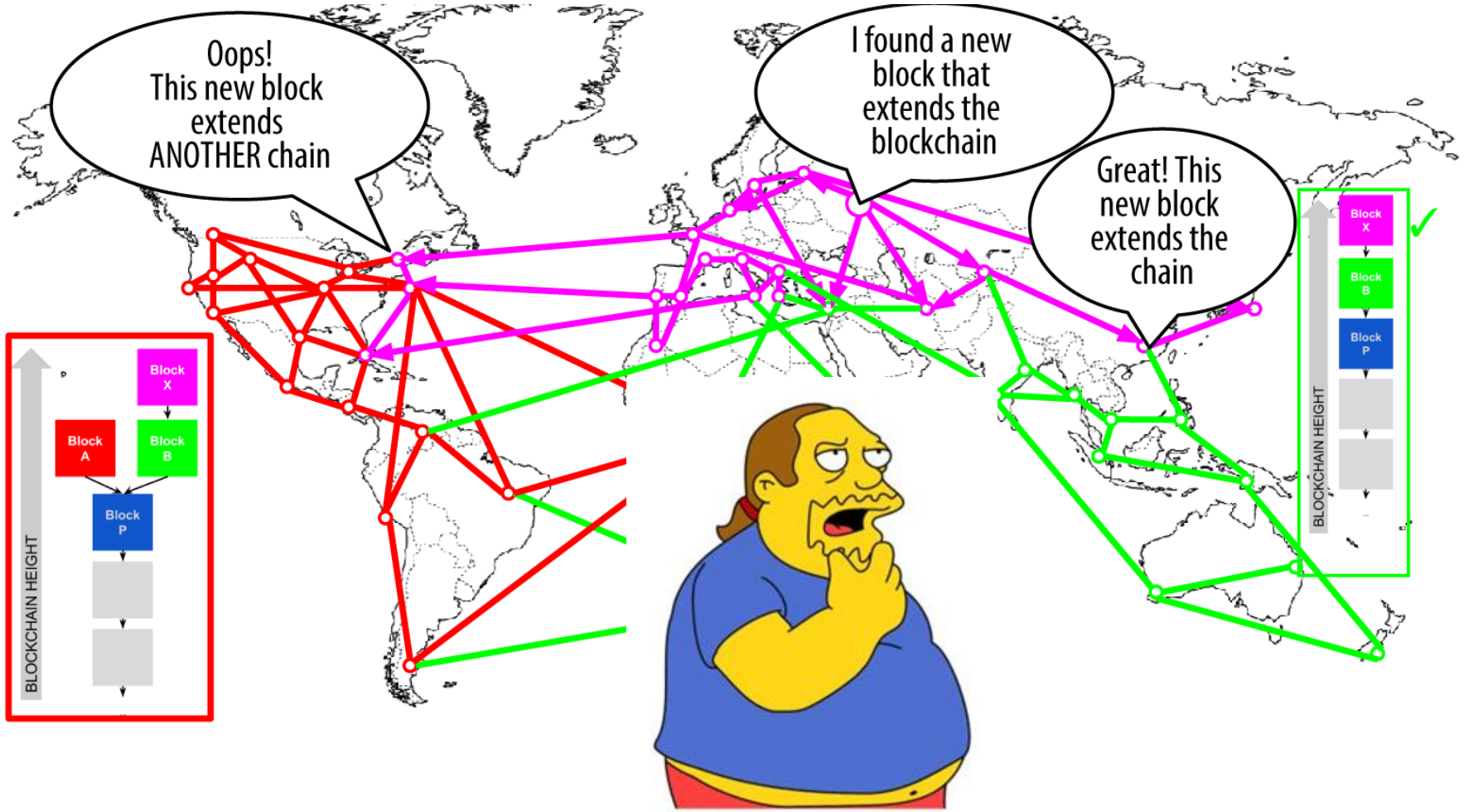
Merkle树 和 Merkle证明

网络中的分布式交易处理

- 在网络中使用去中心化而可信的方式确认交易信息
- 通过区块确认产生比特币



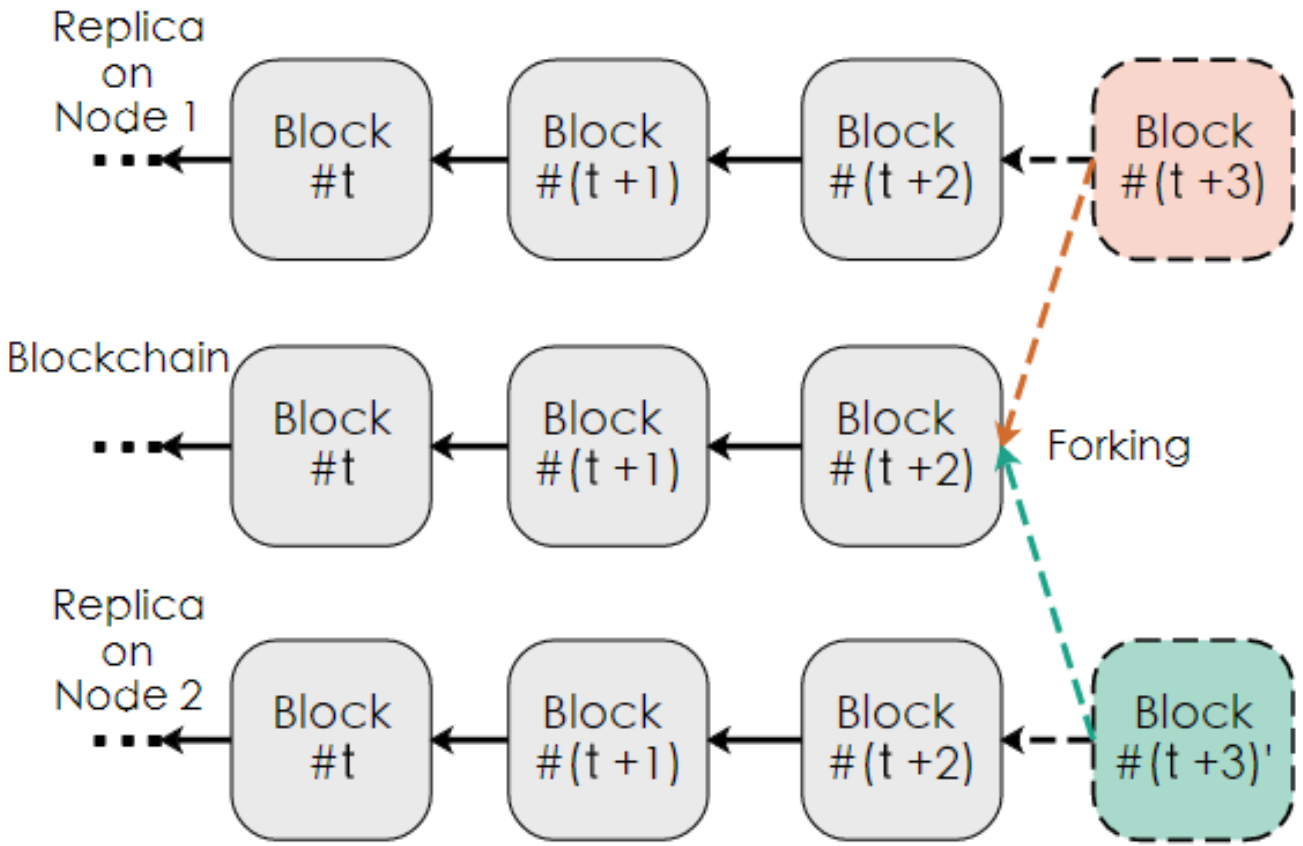
区块链（Bitcoin）网络：一个开放式的（Permissionless）P2P网络



Source: http://chimera.labs.oreilly.com/books/1234000001802/ch08.html

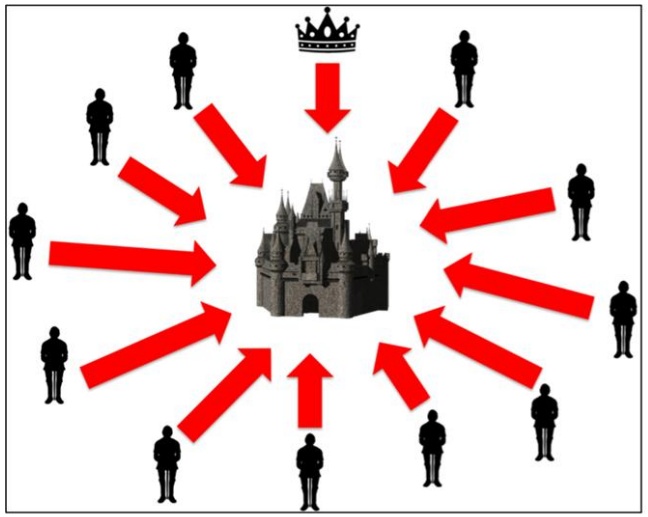
区块链分叉：不同网络位置中的临时状态

- **中本聪（Nakamoto）协议（Proof-of-Work）：**
  - **最早确认的最长链为主链。**

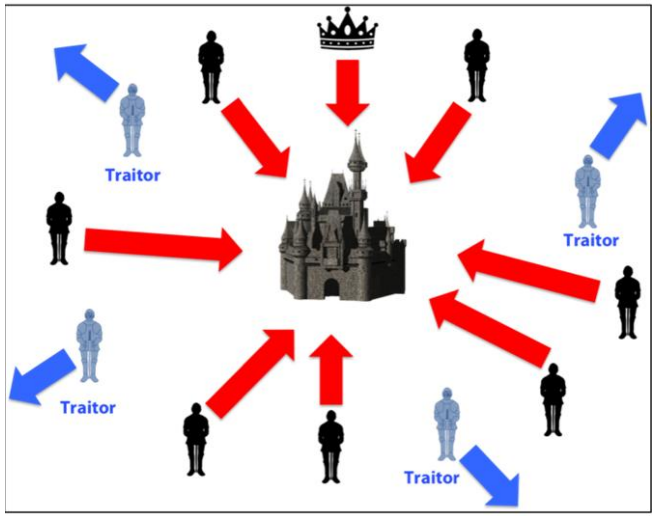


解决分叉的方法：共识协议（Consensus Protocol）

- **共识协议定制问题的产生背景**
  - 在多个分布式节点对系统某项状态（变化）进行确认的过程中，会因为通信错误、节点本身错误甚至是恶意节点的攻击行为，导致节点对系统状态的认知不一致。
- **拜占庭将军问题（Byzantine General Problem）**
  - 中古时期的拜占庭帝国把将军们的部队部署在帝国各驻地，由于通信不便（信使可能被敌人截获、信使叛变等）以及各军队分队的将军也可能叛变，各个军队分队的将军们需要一种共识机制决定是否进攻（“1”）或撤退（“0”）[注]。
- **拜占庭类型错误**
  - 即节点上的任意错误，包括节点死机、节点通信错误、节点说谎等。



Coordinated Attack Leading to Victory

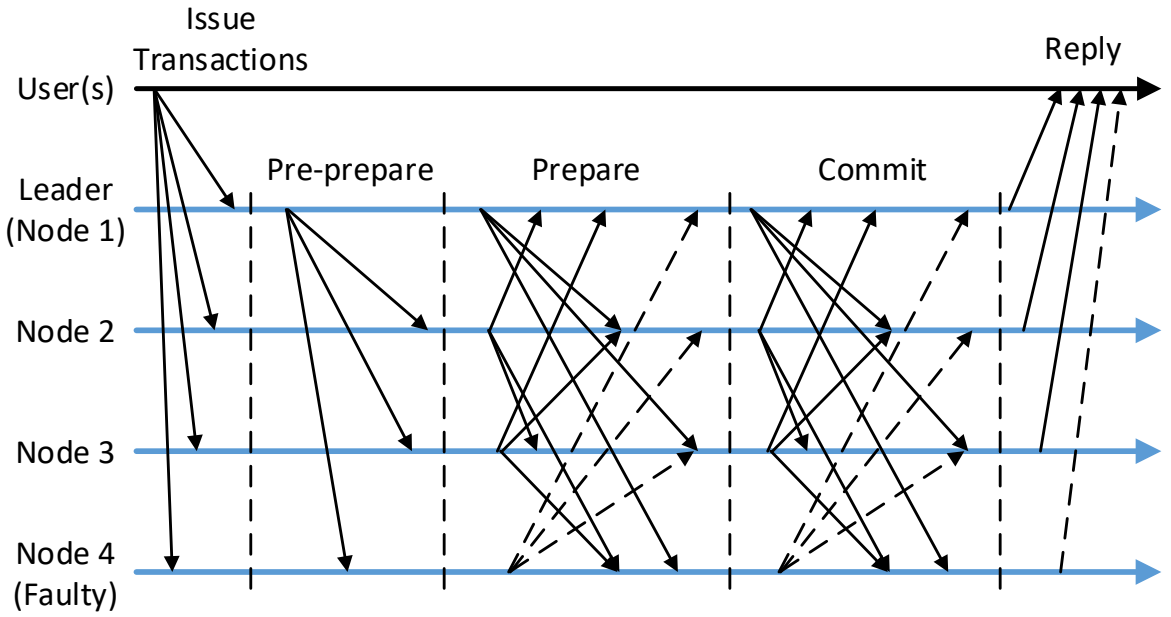


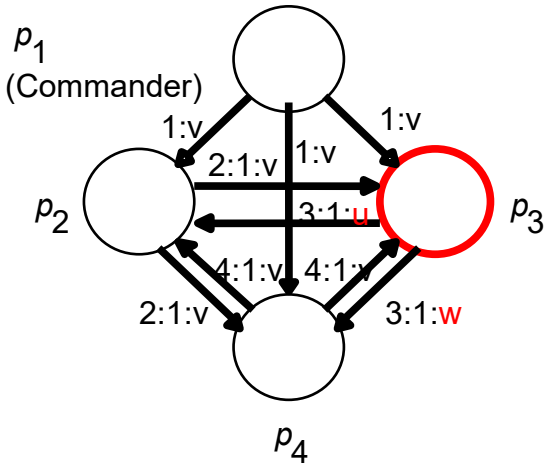
Uncoordinated Attack Leading to Defeat

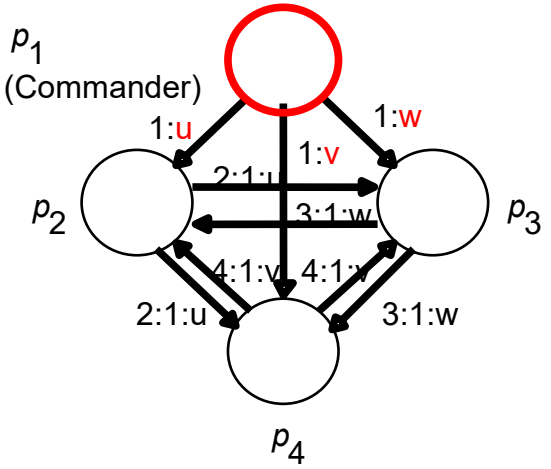
注：此问题是由现代计算机学家Lamport杜撰的。

传统同步条件下的共识机制：拜占庭容错算法（Byzantine Fault Tolerance，BFT Algorithm）

- **BFT算法结构：节点身份**
  - 1主（Master/Leader）：向网络提供一个状态值。
  - *N*从（Slave/Peer/Follower）：通过互相通信决定接受收到的值（状态更新）或拒绝此值。
- **共识流程（见右图上）**
  - （1）Pre-prepare阶段：客户端（通过主节点）广播消息值t（transaction）到各个从节点。
  - （2）Prepare阶段：从节点们收到消息t后，进行消息验证等操作，然后向其他从节点广播自己确信的消息值（Prepare信息）。
  - （3）Commit阶段：一个从节点当收到大多数从节点发送的同一信息t（大多数代表节点数多于网络节点总数的2/3），向其他节点广播一条commit信息。
  - （4）Reply阶段：当节点收到多于2/3的节点发送的验证后信息t，则向客户端发送一条reply信息。
  - 共识条件：假设拜占庭类型节点数为F，则当网络节点总数N中的善意节点多于2F个时（N>3F），BFT算法有效。



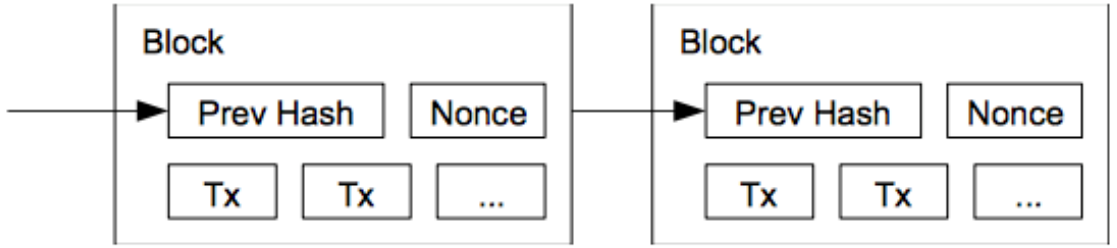


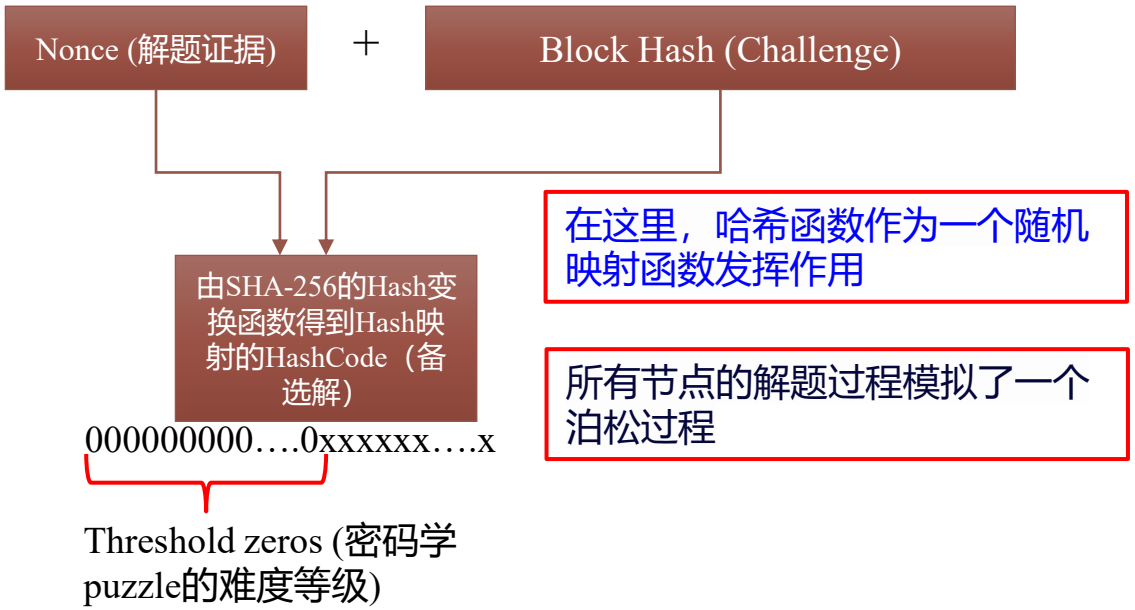


BFT解决拜占庭将军问题：拜占庭错误节点/消息用彩色表示

区块链如何解决拜占庭错误下的共识问题

- **一般情况下的公共区块链：**
  - **网络节点数不确定：随时可能由节点加入或退出当前的共识形成过程。**
  - **节点之间只有松散的同步机制（半同步）。**
- **工作量证明机制（Proof-of-Work，或称中本聪/Nakamoto共识机制）**
  - **使用基于随机数的密码学谜题求解过程模拟共识机制中的随机主节点选举机制，以避免主节点可能的恶意行为。**
  - **使用“最长链原则（The longest-chain rule）”确保大多数良性节点（非拜占庭节点）遵循区块链状态修改协议。**





1. 将当前打包的Transaction的Merkel头、时间戳和上个区块的Hash值组成区块头；
2. 通过迭代选择 **Nonce = Nonce +1**，计算Nonce和区块头的**HashCode**；
3. 迭代终止条件:：**HashCode ≤ Threshold**，即得到满足Threshold条件的解；
4. 对条件中每个多出来的0比特位, 迭代次数的期望值（也就是Proof工作量）将翻倍，例如：$2^5=2^4*2$。

区块链共识算法Proof-of-Work的有效性的理论分析

- **工作量证明算法的有效性由以下论文（及其作者的系列论文）证明：**
  - [Garay2015] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*, Sofia, Bulgaria, Apr. 2015, pp. 281–310.
- **PoW算法和BFT算法的特性的对应关系**

PoW算法特性	对应的拜占庭一致性算法的特性
Common prefix property	Agreement (consistency) and permanent order
Chain quality property	Validity
Chain growth property	Liveness (termination)

- **一致性（协定性）：所有诚实节点的决策都是相同的。**
- **有效性（正确性）：如果所有诚实节点都通过相同的区块来扩展区块链，他们就会同意采用这个新区块。**
- **活性（终止性）：由诚实节点发出的所有交易最终都将得到确认。**

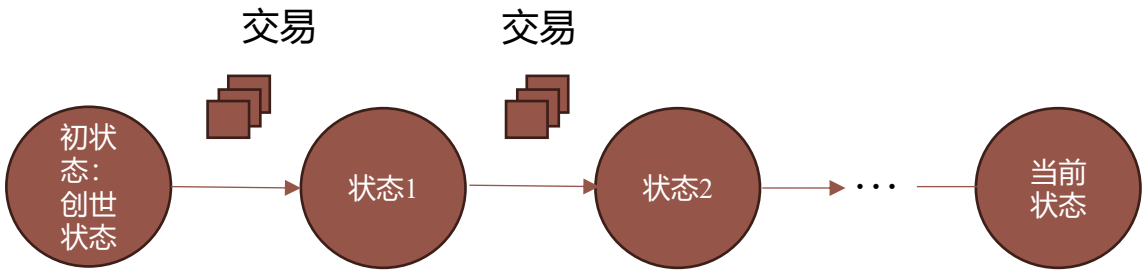
- **PoW算法给区块链研发人员的启发**
  - **"诚实的大多数"（The honest majority）节点遵循最长链原则更新自己的本地区块链状态。**
  - **我们可以设计其他的随机Master节点选择机制，只要保证：$\Pr_i^{\text{win}} = \frac{w_i}{\sum_{j \in \mathcal{N}} w_j}$，即节点胜出概率与节点-网络资源例比相关。**

区块链和物联网的结合：作为一种信息防篡改的分布式（或去中心化）数据-管理-执行中间件

- **本质上，区块链（如Ethereum）是一个基于交易（Transaction）型数据的分布式状态机**
  - **有限状态机一般涉及四个概念：状态（state）、事件（event）、触发事件后的动作（action）和相关的状态变换（transition）.**

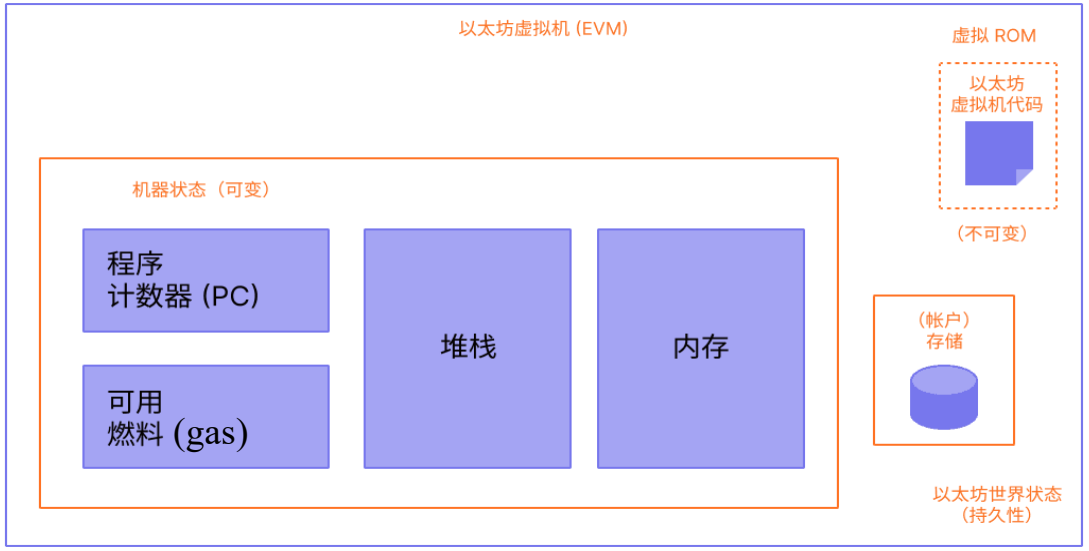
右图：一个简单的门的状态机

- **区块链的状态机**
  - **状态：由区块链中的所有账户的状态组成；**
  - **事件：指区块链中的交易；**
  - **动作：区块链中的转账操作；**
  - **状态的变换：指区块链中交易确认的过程。**

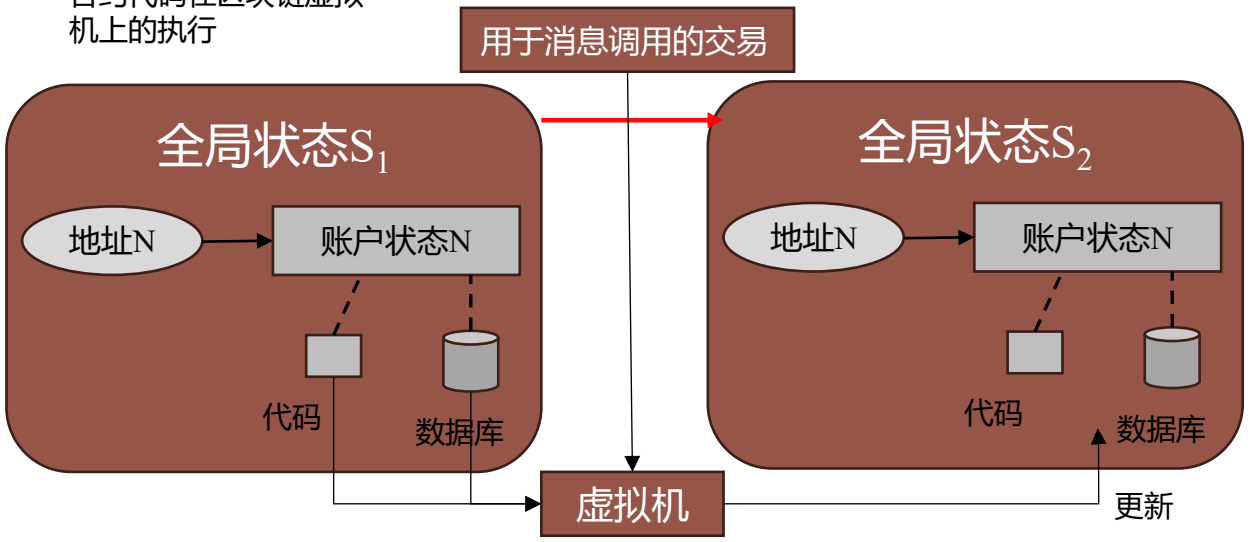


基于区块链状态机的虚拟机

- **区块链的虚拟机（如Ethereum Virtual Machine）**
  - 区块链的全局状态是一个大型数据结构。它不仅保存所有帐户和余额，而且还保存一个机器状态，即完备图灵状态机。它可以根据预定义的一组规则（如智能合约代码）在不同的区块之间进行状态转换，由此执行任意的机器代码。
  - 区块链的虚拟机坐落在区块链存储层协议（如非对称加密协议、Hash协议，数据库协议等）和区块链P2P网络层协议以及共识协议（如PoW）基础之上。
  - 区块链虚拟机的部署主要目的是为了保证链码（或称智能合约）在区块链网络上的不同节点执行时能够得到一致性的结果。
- **智能合约在区块链虚拟机上的执行**
  - 智能合约是一组脚本程序，目的在于实现业务逻辑（链码没有用户界面）；
  - 智能合约运行在区块链虚拟机上，通过API与虚拟机对接；
  - 应用层：人类或机器节点用户通过应用层提供的人机界面或机器-机器间接口使用/调用智能合约。
  - 应用举例：基于以太坊开发的虚拟宠物App， CryptoKitties。



合约代码在区块链虚拟机上的执行



图示来源：https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

智能合约（链码）简介

- **以太坊智能合约的初步部署**
  - 智能合约由某账户（地址）通过发起特定交易的形式部署到以太坊的网络节点。
  - 举例：某账户(0x583031d1113ad414f02576bd6afabfb302140225)希望主动发布编译成字节码的智能合约到网络：

{ "from": 0x583031d1113ad414f02576bd6afabfb302140225,
"nonce": 100,
"input":
"6080604052348015610010576000806040518080f73636973736f72730.....",
“to”: nil ,
"timestamp": 2018-11-27 12:32:48,
"sign": "" ... }

  - 以上代码中，目的地址to为nil（空）指示以太坊共识节点此交易为智能合约发布交易。
  - 智能合约地址由发布账户地址+nonce的Hash值决定： hex(hash(from, nonce)[0:20])

- **以太坊的智能合约构成：ABI+字节码**
  - ABI：Application Binary Interface，即应用程序二进制接口，与被编译成二进制代码（字节码）的程序交互。

ABI由若干个JSON字符串构成，用来表示智能合约的函数和事件，如（注意JSON实际不支持注释）

{ “name”: “sum_of_input”, /*指定函数名*/
“type”: “function”, ”, /*指定sum_of_input的类型为函数*/
“stateMutability”: “pure”, ”, /*指定sum_of_input不访问区块链*/
“inputs”: [”, /*指定sum_of_input的输入参数和类型*/
{
“name": “_max”,
“type": “double“
}], ... }

  - ABI只提供智能合约的定义，还需要智能合约的可执行代码：字节码。
  - 字节码在以太坊节点部署前需要先根据函数名和输入类型的签名值生成函数选择器，再通过调用字节码，生成返回值。

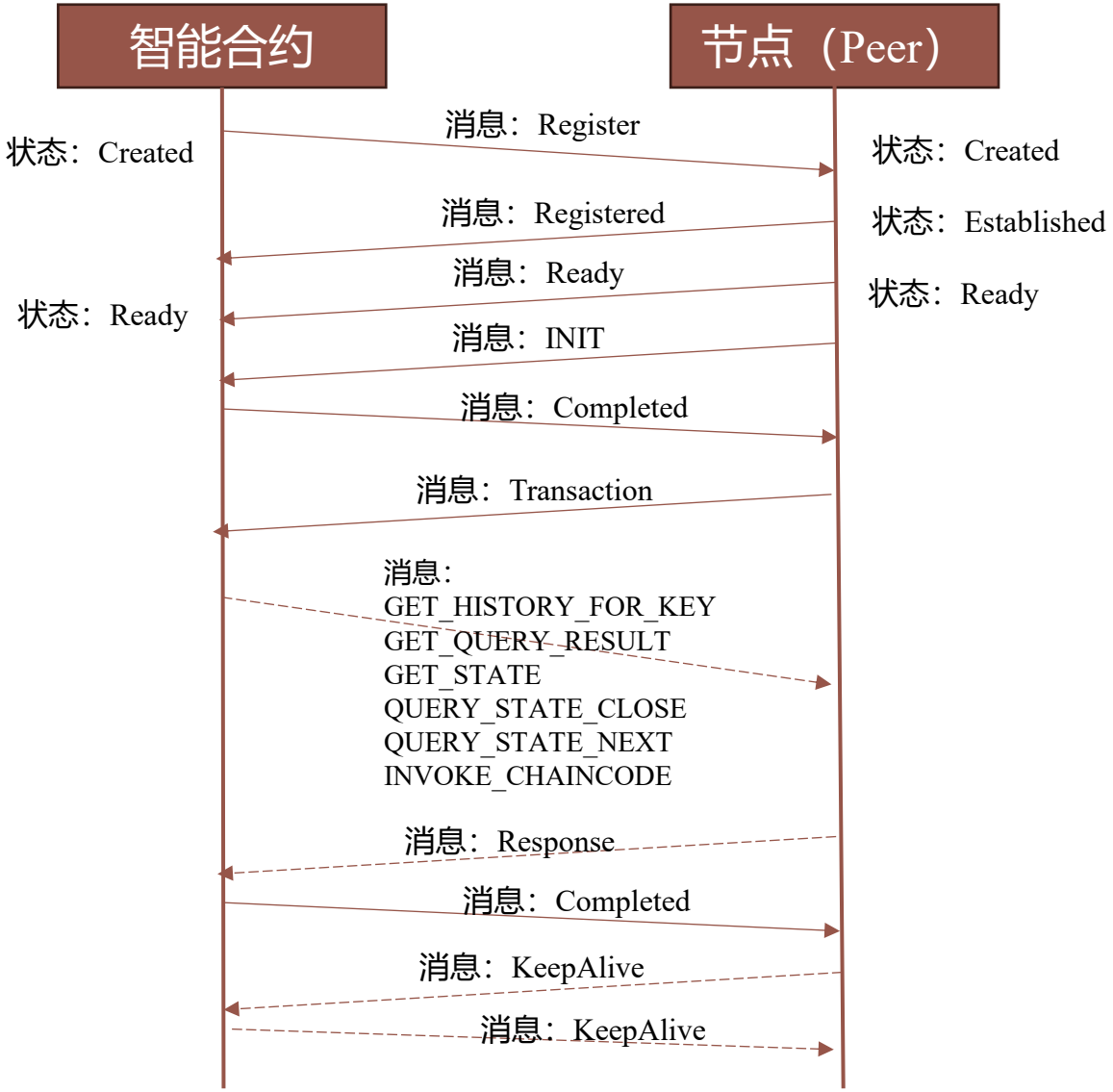
以太坊的智能合约（续）

- 智能合约的全网部署
  - 收到智能合约的共识节点在执行前先验证合约发起方的签名，并验证发起方Token余额是否支持合约执行。
  - 以上验证无误后，共识节点开始执行交易内容。同时广播这个交易到它的所有相邻节点。
  - 相邻节点进行同样的操作，并将交易继续广播，从而，所有以太坊网络节点均会执行此交易，即部署和执行同样的智能合约代码。
- 智能合约的执行
  - 智能合约的调用者通过向合约地址发起一笔交易调用智能合约，举例如下：

    { “from”: “调用者地址”,
    “to”: “合约地址”,
    “value”: 0,
    “input”: “函数选择器+输入参数编码的16进制字节码”,
    “gas”: 驱动共识节点执行合约的Token金额,
    “sign”: “”,
    “nonce”: 交易发起方的最后一次持续交易+1 }

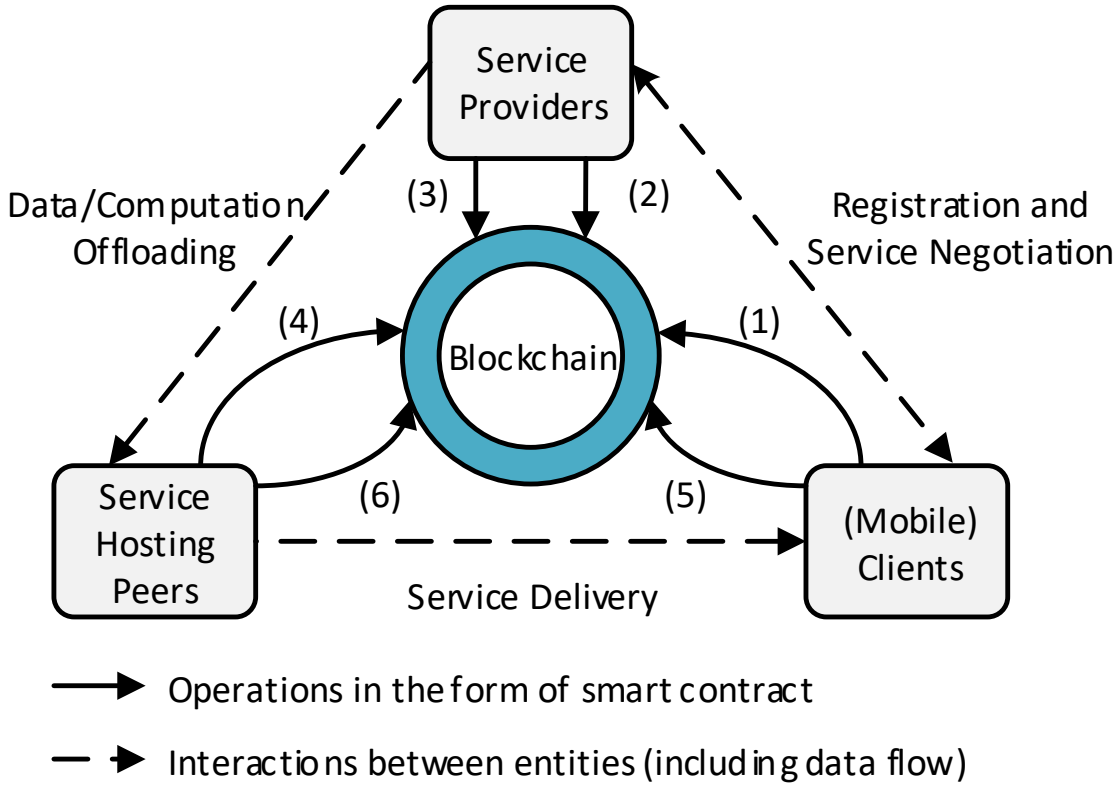
智能合约（链码，ChainCode）简介：Hyper-Ledger Fabric的智能合约

- Fabric的智能合约由于系统节点角色设计与以太坊（Ethereum）有所不同
  - Fabric引入了通道（channel）的概念，每个通道有独立的区块链和实例化智能合约，通道对账本是隔离的。
  - 与Ethereum不同，Fabric的节点（Peer）角色被划分为背书节点（Endorser）和提交节点（Committer）：Endorser负责对交易进行签名，Committer负责维护区块链的状态和副本。
  - Fabric为解决区块链分叉问题还引入了排序节点（Orderer）对客户端提交的交易数据进行排序服务。因而智能合约的部署和调用在经过提交节点外还要额外通过排序节点进行处理。



区块链作为分布式（去中心化）的中阶层和服务通道（Service Tunnel）

- 区块链是否能够作为大规模IoT数据直接处理的中间件?
  - 原始数据上链? ✖（原因：分布式节点保持数据副本的成本过高）
  - 数据实时性能否保证? ✖（原因：除联盟链结构如HyperLedger Fabric外，类似Bitcoin的公共区块链的数据吞吐量不足以支持IoT节点在线实时数据处理的需求）。
- 区块链作为服务和管理信息的通道及中间件
  - 区块链作为多方（Multi-Stakeholder）信息及服务管理平台，使用智能合约提供自组织性的节点间协同服务（典型应用如云-边协同服务管理和CDN服务管理）。
  - 区块链作为IoT数据存证服务平台：提供数据存证的去中心化和安全性、透明性和可追溯性、不可篡改性，以及基于智能合约的存证自动化服务。



作业（第一节）

- 题1：结合课本课件和互联网资料，简述云边协同的常见场景。
- 题2：简述物联网中间件为何产生，以及它的作用、分类分别是什么？（请通过搜索、查阅相关资料，在课件基础上做答。）
- 题3：画出物联网系统数据-信息流处理的“端到云到端”模块图，并尝试解释普通“数据库”和云端“数据池/数据仓库”之间的差别。
- 题4：简述数字孪生中机制建模和数据驱动建模的差别。

作业（第二节）

- 题1：结合课件和互联网资料，简述什么是区块链，以及工作量证明（Proof-of-Work）的基本原理是什么。
- 题2：结合课件和互联网资料，简述什么是对称加密和非对称加密，非对称加密的两类秘钥的作用是什么。
- 题3：结合课件和互联网资料，简述BFT（拜占庭容错算法）的流程并画出流程图。
- 题4：结合课件和互联网资料，简述HTTPS协议下秘钥分发和交换过程，以及有效数据（Payload）的加-解密方法是什么及其原因?

结语（Q&A）

- 参考文献

- [1] 周济等，智能制造导论，高等教育出版社，2021.
- [2] 黄玉兰，物联网概论，人民邮电出版社，2018.
- [3] 郭斌等，智能物联网导论，机械工业出版社，2022.
- [4] 王兴伟等，工业互联网，科学出版社，2024.
- [5] 褚华，霍秋艳主编，软件设计师教程，清华大学出版社，2018.
- [6] 林维锋等，超级账本HyperLedger Fabric区块链开发实战，人民邮电出版社，2020.
- [7] [英]佩里·肖，王颖等译，Java物联网人工智能和区块链编程实战，清华大学出版社，2020.
- [8] Wenbo Wang, D-T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen and D-I, Kim, “A survey on consensus mechanisms and mining strategy management in blockchain networks,” in IEEE Access, vol. 7, pp. 22328-22370, 2019.