互联网发展的三个阶段

web1.0

静态页面，内容只能供用户去阅读，类似于在网络上读报纸或者看书。

web2.0

动态互联网，实现用户之间的互动，比如twitter，facebook，titok等。

web2.0中厂商用免费或极低的成本吸引用户，通过获取到用户的信息来推流广告从而获得利润。

打个比方就是 厂商在一片地上种了很多草，吸引羊来吃，趁着羊吃草的功夫把羊身上的毛薅下来拿去卖钱，而羊自己并不在意这些毛，可以说是一种双向互利的方式。

web3.0

web3.0是一个很模糊的概念，随着区块链技术的发展，基于区块链的web3.0诞生。

接着用上面的例子来说，随着web2.0的发展壮大，稀缺的不再是草，而是羊毛，也就是用户身上的数据。那么羊毛的重要性愈加突出，所以提出web3.0的概念，也就是拥有自己的一片空间，别人无论如何都无法修改，也就是将羊毛（数据）存放在了一个非常安全的地方中，相比于web2.0，不但实现了动态的交互，也实现了数据的“拥有”。

web3的概念非常模糊，可以说是一个大方向，按照我个人的理解可以说是在互联网创造了一个虚拟的世界，这个虚拟的世界拥有和现实世界一样的货币交易系统以及其他体系，能够自主维持运转的这样一个“虚拟生态系统”，而这个生态系统的生存法则就是“去中心化”。

什么是去中心化？

比如现在市面上的app都由一个厂家负责，厂商可以随意删除控制用户数据，形成了以厂商为中心的服务体系，去中心化就是没有中心厂商作为核心，而是所有用户形成一个能够自力更生的体系。

密码货币

随着web2.0发展，数字货币使用越来越多，而在区块链技术的支持下，数字货币也出现了全新的存在形式，去中心化的密码货币，世界上第一种密码货币就是比特币。像纸币有防伪印一样，密码货币通过密码学的散列计算出的hash值并且和智能合约进行绑定，密码货币基于去中心化的机制，与依赖中心监管体系的银行金融系统相对。之后出现的数种密码货币被创造，它们通常被称为altcoins。

区块链

区块链的防篡改机制
一个区块中储存了三样东西：数据，前一个区块的hash值，自身的hash值（由数据和前一区块的哈希值共同决定），如果要更改某一区块的内容，那么该区块（a区块）的hash值就会改变，下一区块（b区块）储存的a区块的hash值无法对应a区块当前哈希值，那么这两个区块间的链接就会断开。

如果想要篡改某一区块的数据，我们就要将这一区块以及后续所有区块的hash值进行重算，比如一条区块链里面有abcde五个区块，当我们篡改了b区块的数据，那么我们就要带着b区块的新hash值和c区块的数据重新计算出c区块的新hash值，然后再带着c区块的新hash值和d区块的数据重新计算d区块的新hash值，再带着d区块的新hash值和e区块的数据重新计算e区块的hash值.....其实在重新计算某一区块的hash值的过程也就相当于创造了一个新的区块，因此篡改一个区块以及后续区块所需的时间取决于创造一个区块所需要的时间。

这个看起来对算力要求似乎非常庞大，但是现代计算机其实是可以做到这一点的，如果我们有一台超大算力的计算机，那么是不是轻松就可以改变区块链的内容了？为了防止这种情况的出现，区块链加入了工作量证明机制（proof of work）简称pow

我们用游戏举例说明一下pow，我们刚才说到用超大算力计算机来篡改区块链，这就好比你拿着满级神装在新手村乱杀，区块链是不允许这种情况出现的，因此它会上调怪物属性，也就是会增加创造一个区块所需的难度，使每一新区块被创造时都保持在十分钟左右（当然这个时间是可以更改的），因此即使是一台超高算力的计算机想要篡改一个区块所需的时间仍然是

创造一个区块的时间□□n min。

那我们所说的挖矿是什么呢？上面提到的情况是想要篡改区块中的数据，那么我没

有恶意，我只是单纯的创造区块去给自己或者他人使用，这个创造区块的过程牺牲了我电脑的算力和一些其他资源，所以作为补偿，创建区块的人会得到密码货币的奖励，这就是我们所说的挖矿。

区块链的点对点网络结构

在传统的web服务中，传统的链接对象基本都是客户端和服务端，众多客户端访问一个服务端来进行交互，而在区块链的点对点网络结构（peer to peer）中，不再有客户端与服务端的概念，每一个节点间相互平等，并且包含完整的区块链数据存储，也就是说每一个节点中都储存了整个区块链网络中的所有信息，这样即使一个节点出现故障，其他所有节点也在帮他记录信息，这些记录了所有节点区块链的节点叫做全节点，当然也有只储存了自己信息的轻节点，比如区块链用来储存转账记录，那么每一节点都储存了所有节点之间的转账记录，每一节点储存的内容也是相同的，如果某一节点与其他节点出现差异，那么该节点或许就有被篡改过的可能了，但是被篡改几乎是不可能发生的，原因看下面。

点对点网络结构下的所有节点拥有判断区块是否被篡改的能力，当一个新区块想要加入某一节点的区块链时，该节点会向其他所有节点进行广播，所有的节点进行判断，如果50%以上的节点都认为该区块没有被篡改，那么这个区块就可以成功的加入区块链当中，反言之如果想要篡改某一区块的数据，你首先要将这一区块后的所有哈希重新计算，并且还要更改超过百分之五十节点的这一区块后的所有区块的哈希，那么就要拥有超过全网50%以上的算力才可以，这付出的代价是相当高的，这就是区块链网络系统的少数服从多数原则。

DAPP

Dapp 是什么？

APP (Application) 指的是手机里的应用程序，像是微信、微博、抖音...等都是日常生活中常会使用到的 App。

而 Dapp 的全名为去中心化应用程序（Decentralized Application），是建立在区块链系统网络上，所提供的服务都具有公开透明、不可篡改的特性。

③ 100+SRC漏洞分析报告
④ 150+网安攻防实战技术电子书
⑤ 最权威CISSP 认证考试指南+题库
⑥ 超1800页CTF实战技巧手册
⑦ 最新网安大厂面试题合集（含答案）
⑧ APP客户端安全检测指南（安卓+IOS）

以下是 Dapp 所具有的要素：

1. 代码开源：程序代码皆公开透明，任何人都可以查阅及审核，避免项目方说到没做到。

2. 分布式帐本：降低数据遗失的风险，且没有任何其他第三方有权能够窜改数据。

3. 数据所有权：除本人（私钥持有者）外，任何人皆无法动用该帐号的数据。

为什么 Dapp 会崛起?

事实上，App 都是中心化的应用服务，用户所使用的数据都会存储在单一服务器系统里，代表公司能掌控用户的所有数据，但相关问题也随之浮出水面。

数据所有权归属问题

用户在 App 上的个人资料、搜索浏览纪录等信息都会存储在中心化系统的服务器里，这也意味着软件公司能够借由这些数据来营利。

也导致像是微博、抖音等企业，能透过搜集的用户数据来投放广告，并借此获利。等于企业能用你的信息来赚钱，但你却分不到任何好处，甚至还可能受到影响（例如被疯狂投放广告、或个人资料被平台外泄）。

另外，传统手游的游戏道具、帐号数据也都属于公司所有，一旦宣布停止营运，这些资产也会随着官方服务器关闭而消失。

但在 Dapp 中，你的游戏道具、帐号都会以 NFT 形式储存在链上，因此只要区块链不倒，你就能持续拥有这些资产。换句话说，Dapp 能够让数据的所有权回归到用户身上。额外提醒，虽然你仍拥有这些资产，但可能会因为游戏已经关闭，导致这些资产的现值趋近于零，你能保有的仍以回忆居多。

过度中心化

App 是由中心化服务器来进行管理，因此企业有时可以专断独行，但用户却没有任何反制的手段：例如可以随意植入广告，或是删除用户的内容、帐号。

而 Dapp 的数据都存在区块链上，因此项目方没办法任意删除用户资料，目前也没有任何广告植入的问题（但不确定未来是否会有项目开始植入广告）。

由于上述几点原因，也让许多人开始对传统的 App 感到不满，于是就有人打算通过区块链“去中心化”的特性来研发能解决上述问题的 App，于是 Dapp 就此诞生。

不过同时也要注意，不是每个 Dapp 都一定符合公开、去中心化的规范，例如 Opensea 就能下架用户的 NFT 和限制用户登陆。

Dapp 与 App 的差异

App 的应用服务是使用中心化服务器，代表软件公司必须要承担存储用户的数据量的营运成本，否则将无法持续地运行。

例如抖音服务器的成本就百万以上，因此必须想办法创造各种营收管道来支持各项支出，像是通过大数据将广告推广到潜在用户面前，借此吸引更多广告商进驻。

而 Dapp 是建立在区块链上，用户在链上进行交易、换币等行为时，是需要自行负担手续费（Gas 费）的，也就代表开发商的运营成本会比传统 App 来得更低（不过有些开发商为了吸引用户，会帮用户负担使用时的手续费）。

接下来我们用代码实现一下简易的POW：

```
const sha256 = require("crypto-js/sha256")function proofOfwo
rk(){    let seed = "y1zh3e7"    let x = 1                //
x?????    while (true){        if(sha256(seed + x).toString(
).substring(0,4) != "0000") // ??????????????????x???seed+x?
???        {
        // ??????????0000?????? ?????????????x??xxxx
    x += 1        }else{            console.log(sha256(seed
+ x).toString())            break        }    }    console.l
og(x)                                        // ??????
????}proofOfwork()
```

实现防篡改机制：

```
/*********************************************????????????
???                                    1.???????hash???????
????hash????  2.???????previousHash????????hash??  *********
*************************************/function validateBloc
k(validBlockchain){    if (validBlockchain.chain.length == 1
)    {        if(validBlockchain.chain[0].myHash != validBlo
ckchain.calcHash())        {            console.log("????")
          return false        }    }else {        for (let
i=1; i<=validBlockchain.chain.length-1; i++)        {
    if(validBlockchain.chain[i].myHash != validBlockchain.c
hain[i].calcHash())            {                console.log(
"????")                return false            }
if (validBlockchain.chain[i].previousHash != validBlockchain
.chain[i-1].myHash)            {                console.log(
"???????")                return false            }        }
    }    console.log("????????????")    return true}
```

将完整的POW整合到区块链当中并实现挖矿功能，最终实现的完整区块链：

```
const sha256 = require('crypto-js/sha256')Date = new Date()c
lass block{    constructor(data,time,previousHash) {
this.data = data        this.time = time        this.nonce =
 1        this.previousHash = previousHash        this.myHas
h = this.calcHash()    }    calcHash() {        return sha25
6(this.data + this.time + this.previousHash + this.nonce).to
String()    }    /**  **/    getAnswer(difficulty){        l
```

```
et answer = ""        for(let i=0; i<difficulty; i++)
 {            answer += "0"        }        console.log(answ
er)        return answer    }   /** ?????? **/    mine(diffi
culty){        let answer = this.getAnswer(difficulty)
  let Hash = this.calcHash()       while(true){            i
f (Hash.substring(0,difficulty) != answer)           {
          this.nonce++               Hash = this.calcHash(
)           }else{               break           }
}       console.log("mine successful!")       console.log(th
is.nonce)       return Hash    }}class blockCahin{    constr
uctor()    {        this.chain = [this.createBlockchain()]
      this.difficulty = 5    }    createBlockchain()    {
     return new block("Genesisblock",Date.toLocaleString(),0
o0000000)    }    getLatestblock()    {        return this.c
hain[this.chain.length - 1]    }    addBlock(newBlock)    {
       newBlock.previousHash = this.getLatestblock().myHash
       newBlock.myHash = newBlock.mine(this.difficulty)
   this.chain.push(newBlock)    }}/************************
**********************????????????? 1.??????hash????????
??hash???? 2.???????previousHash???????hash?? ************
**********************************/function validateBlock(va
lidBlockchain){    if (validBlockchain.chain.length == 1)
 {        if(validBlockchain.chain[0].myHash != validBlockch
ain.calcHash())        {            console.log("????")
       return false        }    }else {        for (let i=1;
 i<=validBlockchain.chain.length-1; i++)        {
 if(validBlockchain.chain[i].myHash != validBlockchain.chain
[i].calcHash())           {               console.log("???
?")               return false           }            if (
validBlockchain.chain[i].previousHash != validBlockchain.cha
in[i-1].myHash)           {               console.log("???
????")               return false           }       }
}    console.log("?????????????")    return true}BlockChain
= new blockCahin()BlockChain.addBlock(new block("this is a t
est",Date.toLocaleDateString(),"anything"))// console.log(Bl
ockChain)// BlockChain.chain[1].data = "????"// BlockChain.c
hain[0].myHash = "0012343566688"// console.log(validateBlock
(BlockChain))
```

数字货币的简单实现

比特币

我们前面说到区块链是用来记录信息的，如果它记录的是转账记录那么它就成了一个账本。

一笔转账信息需要以下四个信息：

付款人 收款人 转账金额 转账时间

我们前面提到了POW，比特币会通过POW将产生一个区块的时间控制在十分钟左右，比特币的工作机制基本如下：

整个区块链是一个网状的网络结构，其中有一个中心，每十分钟发布一个问题（类似于我们之间生成的目标Hash），当问题发布后，该网状网络中的所有结点会来解这个问题（挖矿，爆破目标Hash值），此时就是各结点间的算力比拼，当有一个结点解出该问题后则代表挖矿成功，一个新区块被创造出来，这时该新区块内会自动生成一笔转账记录，其中的收款人就是该区块的挖出者，这时区块链就会自动把奖励发放到收款人的账户上，并且该区块会在整个区块链网络中进行广播，区块链中的每一个结点会对该区块进行验证其合法性，经过验证后该新区块就会被加到区块链上。每四年比特币的奖励会减半一次，最后的比特币总量大约是在2100万个左右。

那么这里会有一个问题，如果过了几年之后，比特币越来越少，每次挖矿后几乎得不到比特币了，那还会有人来挖矿吗？

其实比特币只是比特币区块链中的一个额外奖励机制，整个区块链货币依赖的是每一笔转账记录的手续费，当一个新区块被挖出时那么这个新区块的转账信息（我们刚才说到的比特币奖励机制）就会记录在这个新区块上（以转账的方式发放奖励货币），后续也会记录其他的转账信息，并且会产生手续费，手续费归记录该笔转账信息的区块的挖出者所有。

说到动态调整难度，比特币是怎么调整的呢？

比特币会在每2016个区块诞生后验证一下难度，如果说本来预期中mine这2016个区块所需要的时间是两个星期，而实际只用了一个星期，那么此时比特币区块链就会调整难度，使其达到预期，基本上比特币区块链会每两个星期调整一次难度。

创建自己的数字货币

首先我们要新建一个Transaction类来进行转账记录：

```
class Transaction{    constructor(from, to , amount) {
  this.from = from        this.to = to        this.amount =
amount    }}
```

更改区块中data的含义，此时要记录的是转账信息transaction，并且由于transaction是一个对象，因此在参与计算哈希时要转为字符串（这里将时间改为了Date.now，这样在区块创造时就记录了这笔转账记录的时间）：

```
class Block{    constructor(transaction,previousHash) {
   this.transactions = transaction        this.time = Date.n
ow()        this.nonce = 1        this.previousHash = previo
usHash        this.myHash = this.calcHash()    }    // ??has
h???data???????????data???transaction    calcHash() {
  return sha256(JSON.stringify(this.transactions) + this.tim
e + this.previousHash + this.nonce).toString()    }
```

上面说到奖励货币的发放是通过转账的方式实现的，所以我们在链上实现逻辑：

```
class blockChain{    constructor()    {        this.chain =
[this.createBlockchain()]        this.difficulty = 5
this.transactionPool = []  //?????????        this.mineRewar
d = 50       //???????????    }mineTransaction(minerAddress
)    {        const minerRewardTransaction =  new Transactio
n('', minerAddress, this.mineReward)        this.transaction
Pool.push(minerRewardTransaction)    }
```

之前我们是在外部传入一个区块，在整个货币系统的实现后区块应该是在挖矿时在区块链内部产生的，修改代码：

```
//?Transaction???Transaction Pool?    addTransaction(Transac
tion)    {        this.transactionPool.push(Transaction)
}                mineTransaction(minerAddress)    {        c
onst minerRewardTransaction =  new Transaction('', minerAddr
ess, this.mineReward)        this.transactionPool.push(miner
RewardTransaction)        //??        /*********************
```

```
************************************          * ?????????????
???????????????          * ????????????????????????????????tr
ansaction          * ???????????          ********************
************************************/          const newBlock
= new Block(this.transactionPool,this.getLatestBlock().myHas
h)          newBlock.mine()          //??????????Transaction Poo
l          this.chain.push(newBlock)          this.transactionPo
ol = []     }
```

整个写好的数字代币：

```
const sha256 = require('crypto-js/sha256')class Transaction{
    constructor(from, to , amount) {          this.from = from
        this.to = to          this.amount = amount     }}class
Block{     constructor(transaction,previousHash) {          thi
s.transactions = transaction          this.time = Date.now()
      this.nonce = 1          this.previousHash = previousHash
        this.myHash = this.calcHash()     }     // ??hash???da
ta???????????data???transaction     calcHash() {          retu
rn sha256(JSON.stringify(this.transactions) + this.time + th
is.previousHash + this.nonce).toString()     }     /** ??????h
ash **/     getAnswer(difficulty){          let answer = ""
    for(let i=0; i<difficulty; i++)          {              answ
er += "0"          }          return answer     }    /** ?????? **
/     mine(difficulty){          let answer = this.getAnswer(di
fficulty)          let Hash = this.calcHash()          while(true
){              if (Hash.substring(0,difficulty) != answer)
          {                  this.nonce++                  Hash =
 this.calcHash()              }else{                  break
       }       }       console.log("mine successful!\n")
   console.log("??"+this.nonce+"???????answer?"+Hash)
return Hash     }}class blockChain{     constructor()     {
    this.chain = [this.createBlockchain()]          this.diffi
culty = 4          this.transactionPool = []  //?????????
   this.mineReward = 50          //???????????     }     createB
lockchain()     {          return new Block("Genesisblock",null
)     }     getLatestBlock()     {          return this.chain[thi
s.chain.length - 1]     }     //?Transaction???Transaction Poo
l?     addTransaction(Transaction)     {          this.transacti
```

```
onPool.push(Transaction)    }    mineTransaction(minerAddres
s)    {        const minerRewardTransaction =  new Transacti
on('', minerAddress, this.mineReward)        this.transactio
nPool.push(minerRewardTransaction)        //??        /*****
**********************************************
* ??????????????????????????????        * ??????????????
????????????transaction        * ???????????        ***
**********************************************/
 const newBlock = new Block(this.transactionPool,this.getLat
estBlock().myHash)        newBlock.mine(this.difficulty)
    //?????????Transaction Pool        this.chain.push(newB
lock)        this.transactionPool = []    }}/***************
*********************************????????????? 1.???????has
h???????????hash???? 2.???????previousHash????????hash?? ***
*****************************************/function valida
teBlock(validBlockchain){    if (validBlockchain.chain.lengt
h == 1)    {        if(validBlockchain.chain[0].myHash != va
lidBlockchain.calcHash())        {            console.log("?
???")            return false        }    }else {        for
 (let i=1; i<=validBlockchain.chain.length-1; i++)        {
           if(validBlockchain.chain[i].myHash != validBlockc
hain.chain[i].calcHash())            {                consol
e.log("????")                return false            }
      if (validBlockchain.chain[i].previousHash != validBloc
kchain.chain[i-1].myHash)            {                consol
e.log("???????")                return false            }
     }    }    console.log("?????????????")    return true}Y
1Coin = new blockChain()const Transaction1 = new Transaction
('add1', 'add2', 20)const Transaction2 = new Transaction('ad
d1', 'add2', 5)Y1Coin.addTransaction(Transaction1)Y1Coin.add
Transaction(Transaction2)Y1Coin.mineTransaction("add3")conso
le.log(Y1Coin)console.log(Y1Coin.chain[1].transactions)
```