可验证公平:

WIKI: https://en.wikipedia.org/wiki/Provably_fair

相关讲解文章：https://dicesites.com/provably-fair

Betdice 的 DICE 游戏随机数算法：（本教程目的是介绍验证的详细步骤，并由第三方网站来 hash 或转换所有数据。官方出的验证工具会非常简洁，你不需要重复这里的所有步骤。）

A=客户端浏览器随机种子或用机器人或 Cleos 自定义的种子（官方随后会推出自定义功能）

B=该用户上一局的 hashseed 中冒号后面字符串的 hash 值（如果该账号第一次玩，这个值就是该账户名的 hash 值）

C=A+B=最终的种子（种子实际构造是 A:B）="hashseed"（收据中的）

D=Hash C="hashseedhash"（收据中的）

E=服务器私钥对 D 进行加密签名 //生成一串字符

F=Hash E

G=Hexadicimal （F 的后八位）再转换成十进制数字

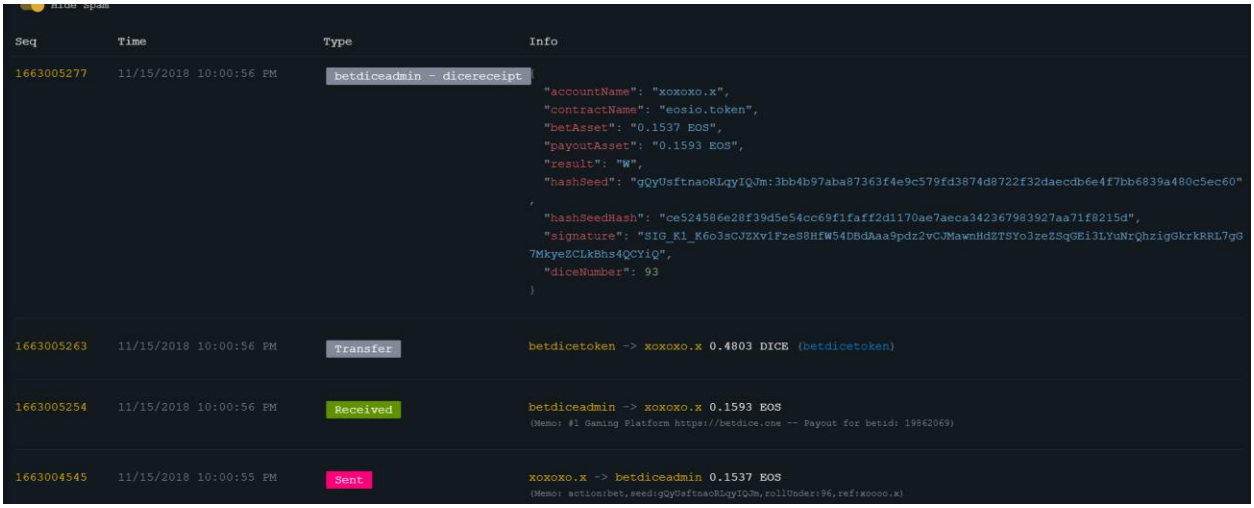
H=MOD(G,100)+1 //取 100 余数后加 1

整个过程关键步骤是 E，私钥加密签名，如果项目方要恶意作弊的话，就要伪造签名加密，伪造签名就无法用公布的公钥验证。

公钥: EOS5bmVmmRRRcdNERtn23Sc4H32e7rGLLXS8hvajUZVfP2PDicBCH

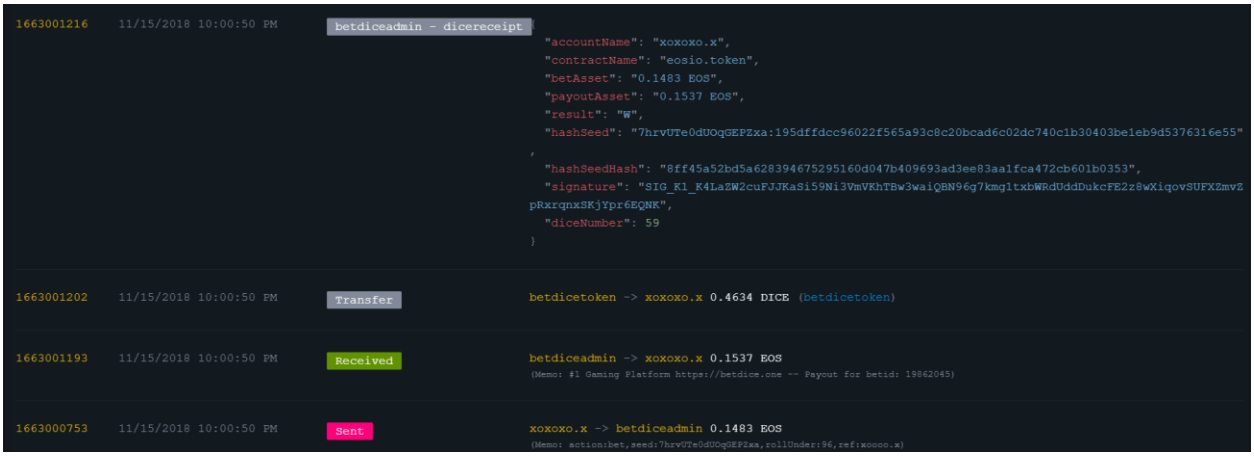
以下是以实际案例举例：随机抽取连续的两个投注:

https://eosflare.io/tx/aa629e973cb44fdc1d10e2c1565dc74c2adc24ed9b6dc640309309ad90279904



```
  "accountName": "xoxoxo.x",
  "contractName": "eosio.token",
  "betAsset": "0.1537 EOS",
  "payoutAsset": "0.1593 EOS",
  "result": "W",
  "hashSeed": "gQyUsftnaoRLqyIQJm:3bb4b97aba87363f4e9c579
fd3874d8722f32daecdb6e4f7bb6839a480c5ec60",
  "hashSeedHash": "ce524586e28f39d5e54cc69f1faff2d1170ae7
aeca342367983927aa71f8215d",
  "signature": "SIG_K1_K6o3sCJZXv1FzeS8HfW54DBdAaa9pdz2vC
JMawnHdZTSYo3zeZSqGEi3LYuNrQhzigGkrkRRL7gG7MkyeZCLkBhs4QC
YiQ",
  "diceNumber": 93
```


https://eosflare.io/tx/f2e1ad182c2f28d886a36e61a6da3c37ed705498f5d9638d62482f6d7c5dd5a3




```
  "accountName": "xoxoxo.x",
  "contractName": "eosio.token",
  "betAsset": "0.1483 EOS",
  "payoutAsset": "0.1537 EOS",
  "result": "W",
  "hashSeed": "7hrvUTe0dUOqGEPZxa:195dffdcc96022f565a93c8
c20bcad6c02dc740c1b30403be1eb9d5376316e55",
  "hashSeedHash": "8ff45a52bd5a628394675295160d047b409693
ad3ee83aa1fca472cb601b0353",
  "signature": "SIG_K1_K4LaZW2cuFJJKaSi59Ni3VmVKhTBw3waiQ
BN96g7kmg1txbWRdUddDukcFE2z8wXiqovSUFXZmvZpRxrqnxSKjYpr6E
```

```
QNK",
  "diceNumber": 59
```

上一次投注结果 59，本次投注结果 93。

步骤如下（所有 hash 都是 sha256）：

A=本次浏览器随机种子= gQyUsftnaoRLqyIQJm

B=上次投注 hashseed 冒号后面字符的 hash 值
=hash(195dffdcc96022f565a93c8c20bcad6c02dc740c1b30403be1eb9d5376316e55)=3bb4b97aba87363f4e9c579fd3874d8722f32daecdb6e4f7bb6839a480c5ec60=本次冒号后面的 hash 值

Hash 转换链接：
https://www.fileformat.info/tool/hash.htm?hex=195dffdcc96022f565a93c8c20bcad6c02dc740c1b30403be1eb9d5376316e55

C：

gQyUsftnaoRLqyIQJm:3bb4b97aba87363f4e9c579fd3874d8722f32daecdb6e4f7bb6839a480c5ec60

D：hash（c）=hash（gQyUsftnaoRLqyIQJm:3bb4b97aba87363f4e9c579fd3874d8722f32daecdb6e4f7bb6839a480c5ec60）= ce524586e28f39d5e54cc69f1faff2d1170ae7aeca342367983927aa71f8215d

Hash 转换链接：（把 c 填到第一个框）

https://www.fileformat.info/tool/hash.htm

E=私钥签名加密（D）=
SIG_K1_K6o3sCJZXv1FzeS8HfW54DBdAaa9pdz2vCJMawnHdZTSYo3zeZSqGEi3LYuNrQhzigGkrkRRL7gG7MkyeZCLkBhs4QCYiQ

F=Hash（E）=
f2eb00b0e083f472da0ca880e0fe9126b474df066fda5265573cb2a6e9a4ec90

G=hexadecimal(e9a4ec90)->十进制=3919899792

转换链接：http://www.statman.info/conversions/hexadecimal.html

H=mod（3919899792，100）+1=92+1=93

可以看出计算的结果和区块浏览器结果是一致的

如何确定服务器用于加密的私钥始终是同一把呢：用 node.js 运行一下代码：

```
const ecc = require('eosjs-ecc');

let signature = '填入 receipt 中的 signature';

let publicKey =
'EOS5bmVmmRRRcdNERtn23Sc4H32e7rGLLXS8hvajUZVfP2PDicBCH';//私钥对应的公钥

let source = '填入 receipt 中的 hashseedhash';

let result = ecc.verify(signature, Buffer.from(source, 'utf8'),
publicKey);

console.log(result);
```

可以任意选取 receipt 验证结果是不是 true，如果是 true，则证明公布的公钥与用来加密签名的私钥是匹配的。

官方会出网页版验证这个步骤，代码会放到 github。

官方验证网址：https://betdice.one/dice/verifier/