底是谁投的,投票人具有隐私性.

4 投票阶段智能合约的逻辑验证

4.1 投票阶段智能合约的模型

我们采用了模型检验工具来验证投票阶段智能合约的内在逻辑,主要验证了步骤 2~步骤 6 以太币流通的内在逻辑,包括:

(1) 投票人根据步骤 2 的合约内容提交保证金和资助金,如果在规定时间内有的投票人没有提交保证金和资助金,则步骤 3 的合约内容能够顺利执行,即已经提交以太币的投票人能够顺利地取回自己的保证金和资助金.

(2) 投票人根据步骤 4 的合约内容提交选票后,可以顺利取回保证金.

(3) 候选人根据步骤 5 的合约内容,获胜者可以顺利领到奖金.

(4) 候选人根据步骤 6 的合约内容,如果在规定时间内有的投票人没有取回保证金,则能够平分没有取回的保证金.

由于上述验证的智能合约只涉及以太币的流通情况,我们把原智能合约中步骤 2 提交选票和步骤 4 提交选票承诺值的过程简化为在步骤 4 直接提交真实选票.这当然意味着我们的验证逻辑不涉及隐私性的验证.但是这样简化之后,对以太币的流通情况并没有任何的修改,所以验证的结果对于原智能合约的以太币流通情况是有效的.事实上,通过逻辑验证,我们确认了 Zhao 等人的比特币投票协议[4]并没有完备地考虑比特币的流通.例如,当诚实的投票人公开承诺值,取回保证金之后,如果有不诚实的投票人没有公开承诺值,则所有投票人的资助金不能被候选人领取,也不能退还投票人.这形成了事实上的资助金“黑洞”.由于矿工可以把交易的输入输出差额作为自己的收入,这在事实上鼓励了矿工不诚实的执行协议,以形成资助金黑洞,获得比特币.本文使用了 Spin 模型,给出 1 个时间进程、5 个投票人进程和 2 个候选人进程来验证以太币的流通情况.

1. 时间进程

时间进程用于模拟智能合约中投票人和候选人在不同的时间段内执行不同操作的行为.我们根据进程的执行时间,保守地选取了 2s 作为一个时间段.具体来说,投票人需要在程序开始执行后 2s 以内完成保证金和奖金的提交,2s~4s 内完成选票的提交,超过 4s 停止计时.注意到,在智能合约的实现上,*Commits*、*Claims*、*proposalClaim* 等函数都有计时器的功能,与该时间进程对应.使用 Promela 描述,时间进程核心代码如下.

```
do
  ::time<=4->
    time=time+1;
  ::else->break;
od
```

2. 投票人进程

投票人进程是投票人允许的所有操作.根据时间进程的不同,投票人在 0~2s、2s~4s 这样两个时间段可以执行 3 个不同的原子操作.原子操作在 Spin 中用关键字 *atomic* 标识,标识范围内的代码执行时不可分割.我们设置了 5 个投票人,每个投票人用其进程号减去 1 的值(*pid*−1)来标识,可以看作不同投票人的编号.对每一个投票人,我们设置了 *usrmoney* 来表示该投票人的账户余额,初始化每个账户为 11,设置了 *Contractmoney* 代表智能合约的余额.注意,在在智能合约中,投票人的对手方是智能合约.与投票协议相关的变量包括 *rights*、*voted* 和 *votersNum*,其中,*rights* 表示用户有一次的投票权,*voted* 表示投票者是否提交了保证金和资助金到智能合约,而 *votersNum* 则统计了提交保证金和资助金到智能合约的人数,分别与智能合约的状态变量一一对应.

在 0~2s,投票人的原子操作如下所示,对应实现了智能合约步骤 2 的资金流向和投票状态的改变.

```
::(timer<=2 && rights[_pid−1]==1 && userMoney[_pid−1]>=11)->atomic{
  contractmoney=contractmoney+11;
```

```
    userMoney[_pid−1]=userMoney[_pid−1]−11;
    rights[_pid−1]=0;
    voted[_pid−1]=1;
    votersNum=votersNum+1;
  }
```

在2s后,投票人的原子操作分两种情况:一种是*Commits*顺利执行的情况,对应*Claims*函数;一种是*Commits*阶段有投票人没有参与的情况,对应*stopVoting*函数.这两种情况涵盖了智能合约步骤3和步骤4的资金流向和投票状态的改变.

在有投票人没有参与*Commits*函数时,*votersNum*小于投票人的总数,执行下面的操作,实现了*stopVoting*相同功能.

```
  ::(timer>2 && votersNum<5 && voted[_pid−1]==1)->atomic{
    contractmoney=contractmoney−11;
    userMoney[_pid−1]=userMoney[_pid−1]+11;
    voted[_pid−1]==0;
  }
```

同样在2s~4s,当所有投票人都参与了*Commits*函数时,投票人可以执行*Claims*函数.该函数对应的操作如下.

```
  ::(timer>2 && timer<=4 && votersNum==5 && claimed[_pid−1]==0)->atomic{
    if
      ::voteId=0;
      ::voteId=1;
    fi
    count[voteId]=count[voteId]+1;
    contractmoney=contractmoney−10;
    userMoney[_pid−1]=userMoney[_pid−1]+10;
    claimed[_pid−1]=1;
    backNum=backNum+1;
    break;
  }
```

其中,*voteId*代表随机选择的候选人,*count*[*voteId*]用来统计候选人的投票总数,这两个状态量表示了简化的投票过程;*claimed*标记是否已参与*Commits*函数,用*backNum*统计取回保证金的人数,与*Claims*函数的*backNum*对应.

3. 候选人进程

候选人进程对应候选人可以运行的所有操作.从时间进程上看,候选人的操作时间有两个起点:一个是2s~4s内投票人完成投票后,候选人可以开始确认胜利者和获得奖金;一个是4s后依旧有投票人没有投票,候选人从4s开始可以平分投票人的保证金.第1种情况使用*backNum*这个记录返回保证金的状态量来标识,投票人在2s~4s内完成投票取回保证金,那么这个状态量是5;第2种情况使用了时间和*backNum*状态量以及*votersNum*状态量来标识,其中,*votersNum*表示*Commits*阶段成功执行了,是候选人平分保证金的前提条件.

投票人完成投票时,候选人的操作如下所示,对应*winningProposal*函数和*Prize*函数.

```
  ::backNum==5->atomic{
    contractmoney=contractmoney−5;
    if
```

```
        ::(count[0]>count[1])->
        moneyA=moneyA+5;
        ::else->
        moneyB=moneyB+5;
      fi
      break;
    }
```

其中,*count* 是在投票操作时记录的投票.在测试例中,每个投票人要么选 0 要么选 1,不存在平票的情况.奖金的流向也是要么给候选人 *A* 的账户 *moneyA*,要么给候选人 *B* 的账户 *moneyB*.

如果投票人完成了承诺却没有完成投票,时间也大于 4s 了,候选人就可以平分投票人的保证金.对应 *proposalClaim* 函数,其中,*getDepositA* 和 *getDepositB* 表明候选人取得了保证金.

```
    ::(votersNum==5 && timer>4)->atomic{
      if
        ::backNum<5
          for (i:0,...,4){
          contractmoney=contractmoney−1;
          userMoney[i]=userMoney[i]+1;
          if
          ::claimed[i]==0
            contractmoney=contractmoney−10;
            moneyA=moneyA+5;
            moneyB=moneyB+5;
          ::else->skip;
          fi
          }
        getDepositA=true;
        getDepositB=true;
        break;
        ::else->break;
      fi
    }
```

4. 逻辑验证

前面我们定义了投票人、候选人、时间进程,这些进程运行之后模拟智能合约投票阶段的运行,运行的结果可以反映资金的可能流向.我们使用 LTL 描述期望的运行结果,如果实际的运行结果与期望的运行结果相同,则表明资金的流向满足我们的期望;否则,表明资金流向存在问题.鉴于投票人进程运行的代码是完全一致的,我们在验证资金流向时,仅选取了一个下标为 0 的投票人,即第 1 个投票人的进程.

我们期望验证的 4 个资金流向逻辑描述如下.

- *ltl backMoney*{[·](*votersNum*<5 && *voted*[0]==1)-><>(*userMoney*[0]==11))}

该逻辑定义为 *backMoney*.投票人根据步骤 2 的合约内容提交保证金和资助金.如果在规定时间内有的投票人没有提交保证金和资助金,根据投票人的定义,即 *votersNum*<5.此时,已经提交保证金和资助金的投票人根据投票人的定义,*voted*[0]为 1.那么,这样的投票人应该可以取回保证金和资助金,即可以期望 *userMoney*[0]为 11.

- *ltl IfrightsThenbackDeposit*{[·](*claimed*[0]==1-><>(*userMoney*[0]==10))}

该逻辑定义为 *IfVotedThenbackDeposit*.根据投票人的定义,如果投票人完成了投票,那么 *claimed*[0]应该为 1.此时,投票人应该同时取回了保证金,所以期望账户余额 *userMoney* 应该是 10.对应智能合约的 *Claims* 函数的资金流向.

- *ltl IfSuccessThenOneGetPrize*{[·](*backNum*==5-><>(*moneyA*==5 && *moneyB*==5))}

该逻辑定义为 *IfSuccessThenOneGetPrize*.根据候选人的定义,如果 *backNum*==5 即投票成功了.此时,可以期望候选人获得了奖金,则此时获胜者账户余额应该为 5.

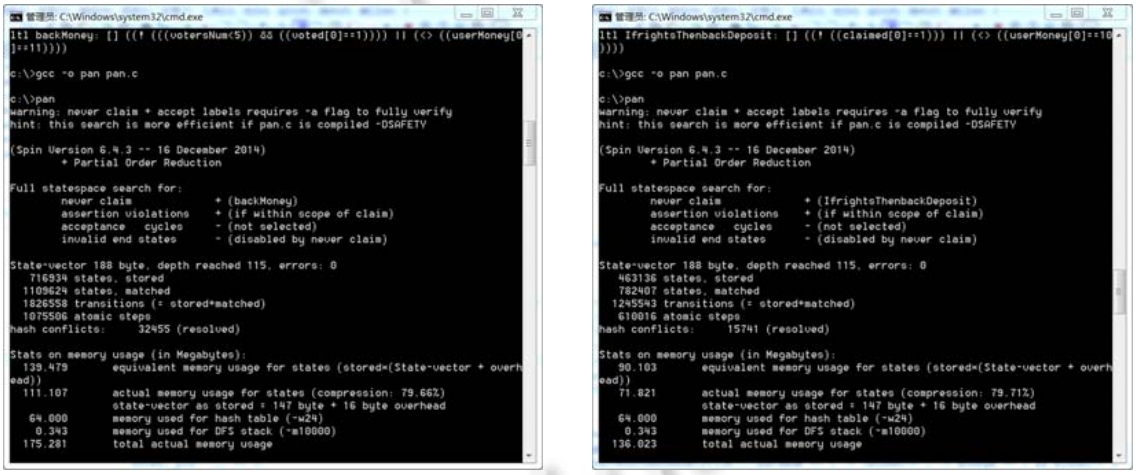
- *ltl IfFaildThenGetDeposit*{[·]((*votersNum*==5 && *backNum*<5)-><>(*getDepositA*==1 && *getDepositB*==1))}

该逻辑定义为 *IfFaildThenGetDeposit*.根据候选人的定义,若所有投票人交了保证金和资助金(*votersNum*==5),但是有投票人没有取回保证金(*backNum*<5),那么可以期望候选人 *A* 和 *B* 平分投票人没有取回的保证金,即 *getDepositA* 和 *getDepositB* 都是 1.

5. 逻辑验证结果

我们把第 3 节的各种进程的定义和第 4 节的逻辑验证形成了 4 个 ballotContract.pml 文件,每个文件中包含完整的进程定义和 1 个单独的逻辑验证条件.对于每一个逻辑验证条件,在 Windows 平台的命令行界面下运行 6.4.3 版本的 Spin 程序,执行 spin -a ballotContract.pml,生成 c 语言的模型检测程序 pan.c,并在命令行中显示该代码所检测的逻辑验证条件.然后执行 gcc -o pan pan.c 命令编译,生成可执行程序 pan.该 pan 程序运行时间进程、投票人进程、候选人进程,尝试所有可能的状态,以判断投票人进程和候选人进程能否满足其逻辑验证条件.

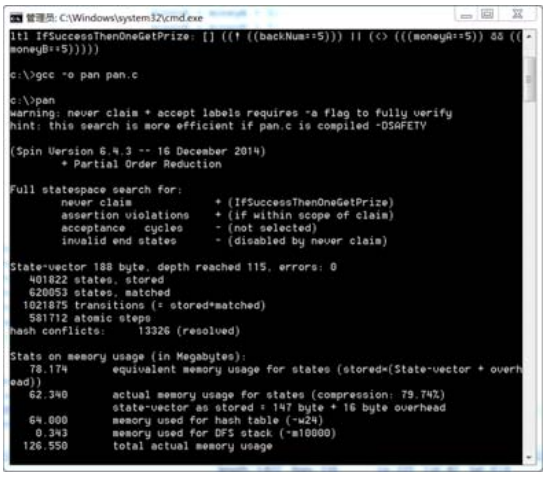
如图 8 所示,4 个逻辑验证条件的运行结果是类似的,只有相关状态和运行序列的不同,而遍历的状态总数是确定的,结论也是相同的,即满足逻辑验证条件(*errors*=0).



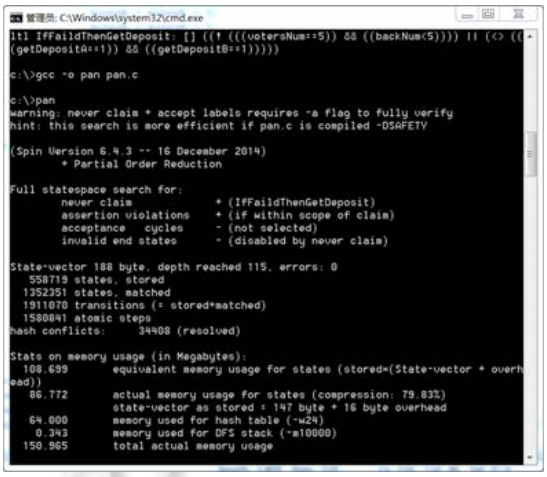
(a) *backMoney* 逻辑验证结果 (b) *IfrightsThenbackDeposit* 逻辑验证结果

Fig.8 Logic verification result of the smart contracts

图 8 智能合约逻辑验证结果



(c) *IfSuccessThenOneGetPrize* 逻辑验证结果



(d) *IfFaildThenGetDeposit* 逻辑验证结果

Fig.8 Logic verification result of the smart contracts (Continued)

图 8 智能合约逻辑验证结果(续)

5 结 论

本文给出了一个保持隐私的以太币投票协议,特别是其投票阶段的智能合约.我们给出了在以太坊中智能合约的详细实现过程,并给出了测试结果.进一步地,我们对智能合约和实现的代码进行了逻辑验证,确保智能合约的资金流向是符合预期的.我们同时指出了 Zhao 等人给出的比特币投票协议因为缺乏逻辑验证的过程,存在资金流向黑洞的问题.

References:

[1] Tian HB, He JJ, Fu LQ. A privacy preserving fair contract signing protocol based on block chains. Journal of Cryptography, 2017, 4(2):187−198 (in Chinese with English abstract).

[2] Wood DG. Ethereum: A secure decentralised generalised transaction ledger homestead. 2014. http://gavwood.com/paper.pdf

[3] EtherCasts. A curated collection of decentralized apps. 2017. http://dapps.ethercasts.com

[4] Zhao Z, Chan TH. How to vote privately using bitcoin. In: Qing S, Okamoto E, Kim K, Liu D, eds. Proc. of the 17th Int'l Conf. on Information and Communications Security (ICICS 2015). Beijing: Springer Int'l Publishing, 2015.

[5] Sasson EB, Chiesa A, Garman C, Green M, Miers I, Tromer E, Virza M. Zerocash: Decentralized anonymous payments from bitcoin. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. IEEE, 2014. 459−474.

[6] Barber S, Boyen X, Shi E, Uzun E. Bitter to better—How to make Bitcoin a better currency. In: Keromytis AD, ed. Proc. of the 16th Int'l Conf. on Financial Cryptography and Data Security (FC 2012). Berlin, Heidelberg: Springer-Verlag, 2012. 399−414.

[7] Andrychowicz M, Dziembowski S, Malinowski D, Mazurek L. Fair two-party computations via Bitcoin deposits. In: Böhme R, Brenner M, Moore T, Smith M, eds. Proc. of the Workshops on Financial Cryptography and Data Security (FC 2014), BITCOIN and WAHC 2014. LNCS 8438, Berlin, Heidelberg: Springer-Verlag, 2014. 105−121.

[8] Kulyk O, Volkamer M. Efficiency comparison of various approaches in E-voting protocols. In: Clark J, *et al*. eds. Proc. of the FC 2016 Workshops. LNCS 9604, Berlin, Heidelberg: Springer-Verlag, 2016. 209−223.

[9] Shahandashti SF, Hao F. DRE-ip: A verifiable E-voting scheme without tallying authorities. In: Proc. of the ESORICS. LNCS 9879, Springer-Verlag, 2016. 223−240.

[10] Sasson EB, Chiesa A, Garman C, Green M, Miers I, Tromer E, Virza M. Zerocash: Decentralized anonymous payments from bitcoin. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. IEEE, 2014. 459−474.

[11] Bentov I, Kumaresan R. How to Use Bitcoin to Design Fair Protocols. Berlin, Heidelberg: Springer-Verlag, 2014. 421−439.

[12] Kiayias A, Zhou HS, Zikas V. Fair and robust multi-party computation using a global transaction ledger. In: Fischlin M, Coron JS, eds. Proc. of the Advances in Cryptology 35th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2016). Berlin, Heidelberg: Springer-Verlag, 2016. 705−734.

[13] Andrychowicz M, Dziembowski S, Malinowski D, Mazurek L. Secure multiparty computationson Bitcoin. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. IEEE, 2014. 443−458.

[14] Decker C, Wattenhofer R. Bitcoin Transaction Malleability and MtGox. Cham: Springer Int'l Publishing, 2014. 313−326.

[15] Buterin V. Ethereum: A next-generation smart contract and decentralized application platform. 2014. https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper

[16] Lin HM, Zhang WH. Model checking: Theories, techniques and applications. Acta Electronica Sinica, 2002,30(12A):1907−1912 (in Chinese with English abstract).

[17] Hu K, Bai XM, Gao LH, Dong AQ. Formal verification method of smart contract. Ruan Jian Xue Bao/Journal of Software, 2008, 19(7):1565−1580 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1565.htm [doi: 10.3724/SP.J.1001.2008.01565]

[18] Barnat J, Brim L, Stříbrná J. Distributed LTL model-checking in SPIN. In: Dwyer M, ed. Proc. of the SPIN 2001. LNCS 2057, Heidelberg: Springer-Verlag, 2001. 200−216.

[19] Liu J. Research and design of electronic voting protocol based on Internet [Ph.D. Thesis]. Fuzhou: Guangxi University, 2001 (in Chinese with English abstract).

[20] He ZH, Ceng QK. Research on LTL attribute decomposition method based on SPIN. Computer Applications and Software, 2014, 31(7):43−46 (in Chinese with English abstract).

[21] Luu L, Chu DH, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Proc. of the 2016 ACM CCS. ACM, 2016. 254−269.

[22] McCorry SSP, Hao F. A smart contract for boardroom voting with maximum voter privacy. In: Kiayias A, ed. Proc. of the Financial Cryptography and Data Security 2017. LNCS 10322, Springer-Verlag, 2017. 357−375.

[23] Tarasov P, Tewari H. Internet voting using zcash. 2017. https://eprint.iacr.org/2017/585

[24] 2017. http://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html#a-simple-smart-contract

附中文参考文献:

[1] 田海博,何杰杰,付利青.基于公开区块链的隐私保护公平合同签署协议.密码学报,2017,4(2):187−198.

[16] 林惠民,张文辉.模型检测理论、方法与应用.电子学报,2002,30(12A):1907−1912.

[17] 胡凯,白晓敏,高灵超,董爱强.智能合约的形式化验证方法.软件学报,2008,19(7):1565−1580. http://www.jos.org.cn/1000-9825/19/1565.htm [doi: 10.3724/SP.J.1001.2008.01565]

[19] 刘峻.基于 Internet 的电子投票协议的研究与设计[博士学位论文].福州:广西大学,2001.

[20] 贺志宏,曾庆凯.基于 SPIN 的 LTL 属性分解方法研究.计算机应用与软件,2014,31(7):43−46.

付利青(1992－),女,湖南郴州人,硕士,主要研究领域为区块链技术及其应用.

田海博(1979－),男,博士,副教授,主要研究领域为安全协议设计与分析,区块链技术.