基于智能合约的隐写术

刘九良，付章杰，孙星明

(南京信息工程大学 计算机与软件学院，江苏 南京 210044)

通讯作者：付章杰, E-mail: fzj@nuist.edu.cn

摘 要： 正在兴起的区块链技术可以使任意节点维护一个权威的、可信的、公开的、不可篡改的分布式账本。区块链本质上是一个全球的、分布式的、防篡改的数据库，它具有去中心化，区块数据基本不可篡改、安全性等特征。区块链的点对点网络可以作为一个安全的通信渠道，可以在节点互不信任的情况下传递信息。因此，区块链是一个天然的适合信息隐藏的载体媒介。基于区块链的信息隐藏可以充分利用区块链的防篡改性，同时又可以将区块链网络作为一个安全的通信渠道。本文提出了一个新的基于区块链的隐写方案，它通过部署含秘智能合约到区块链上实现信息隐藏。我们首先建立了区块链隐写的基本框架，然后提出了基于智能合约源码和基于智能合约字节码两类隐写方案。基于智能合约源码的隐写方案在合约源码中嵌入秘密信息，它需要部署智能合约后公开合约源码，隐藏容量至少可以达到 0.28%，这与普通的文本信息隐藏相近，但它有更高的鲁棒性和更安全的通信渠道；而基于智能合约字节码的隐写方案在编译生成的字节码中隐藏信息，它不公开合约源码，隐藏率至少可以达到 1.75%，并且同样具有鲁棒性强和通信渠道安全的优点。

关键词： 隐写术；区块链；智能合约；鲁棒性；字节码；以太坊

A New Steganography Based on Smart Contracts

Liu Jiu-Liang , Fu Zhang-Jie, Sun Xing-Ming

(College of computer and software, Nanjing University of Information Science & Technology, Nanjing 210044, China)

Abstract: The emerging blockchain technology enables any node to maintain an authoritative, reliable, open, tamper-proof and distributed ledger. The blockchain is essentially a global, decentralized, and tamper-proof database, which has the basic characteristic of decentralization, security and tamper-proofing. The p2p network of the blockchain can be used as a secure communication channel to deliver information when nodes do not trust each other. Therefore, the blockchain is a natural carrier medium which is suitable for covert communication. Information Hiding based on the blockchain can take full advantage of the tamper-proofing of the blockchain. And the blockchain can be used simultaneously as a secure communication channel. In this paper, a new steganographic scheme by deploying smart contracts to the blockchain is proposed. First, we establish a basic framework of steganography based on blockchain, and then propose two types of steganographic scheme based on source code and bytecode of smart contracts. The steganographic scheme based on the source code of smart contracts embeds secret information in the source code of contract. This scheme needs to open the source code of contracts. The hidden capacity of this scheme can reach at least 0.28%, which is similar to the ordinary text steganography. However, it has stronger robustness and a safer communication channel. The steganographic scheme based on the bytecode of contracts embeds the information in the compiled binary code. By this scheme, the hiding rate can reach 1.75% and the source code of contracts does not need to be opened. This scheme also embodies the advantages of high robustness and secure communication.

Key words: steganography; blockchain; smart contract; robustness; bytecode; ethereum

1 介绍

区块链技术已经在金融、医疗、物联网、房地产和保险等行业有着广泛的应用。区块链是一个分布式的计算架构，其中每个网络节点执行并记录相同的被分组打包到区块中的交易，它可以在没有中心机构的情况下保证每个参与方维护一个安全的、永久的、不可篡改的交易账本[1]。区块链中的每个全节点（完整的区块链客户端）都需要验证每笔交易和每个区块的合法性。每个节点会对区块链进行独立选择，在工作量证明的机制下选择累计工作量最大的链。由于区块链的共识机制，最终所有节点承认一个唯一的、可信的、公开的账本。根据区块链的共识机制，篡改区块数据必须拥有超过全网 50%的算力，这在现实中是几乎不可能实现的。区块链作为一个公开的分布式账本，发送交易的同时可以传递消息，所以可以作为一个点对点的、安全的通信渠道。由于应用广泛，区块链中每天都会涌入大量的数据，这为信息的隐蔽传输提供了保障。基于区块链的隐写术可以充分发挥区块链的防篡改优势和区块链网络的通信优势。因此区块链是一个天然的、适合信息隐藏的载体媒介。

信息隐藏是指将秘密信息隐蔽地嵌入到载体媒介中，从而达到隐蔽通信或者保护版权的目的。在信息隐藏领域中，如何提高鲁棒性、抗检测性和隐藏容量一直是信息隐藏技术的主要目标。目前大多数信息隐藏技术以图片、音频、视频、文本、网页、程序等作为载体媒介[2]-[6]。虽然利用这些载体媒介的信息隐藏技术已经趋近成熟，但是由于这些载体媒介容易被攻

击者恶意篡改，它们的鲁棒性很难达到或者超过区块链信息隐藏的鲁棒性。为了提高传统载体媒介的鲁棒性，信息嵌入算法通常考虑在一些攻击方式下提高秘密信息的提取率。常见的攻击方式有噪声攻击、压缩攻击、表达攻击、几何攻击等。由于区块链技术的防篡改性，这些攻击方式在区块链中几乎不可能实现，所以基于区块链的信息隐藏具有更高的鲁棒性。以图片、音频和视频等为载体媒介的信息隐藏没有一个天然的通信渠道[7]-[9]，因此传递信息不够方便。虽然以网页、论坛和推特等为载体的信息隐藏拥有天然的通信渠道[10]-[12]，但这些载体媒介往往依赖于一个中心机构，这为信息的可靠传递带来了安全隐患。区块链的点对点网络为信息的传递提供了一个较安全的通信渠道。由于区块链的分布式共识机制，在没有中心机构和互不信任的情况下人们同样可以可靠地传递信息。因此基于区块链的信息隐藏可以很好地解决了信息隐藏技术中的鲁棒性和通信渠道问题。

目前我们已经进入区块链 2.0 时代，其中第一个出现的图灵完备的分布式智能合约系统以太坊是这个时代的代表[13]。目前已经出现了大量的智能合约应用，例如市场预测[14]、供应链溯源[15]、股权众筹[16]和证券交易[17]等。因此每天都有大量的信息以智能合约的形式被部署到区块链上，这为利用智能合约进行隐蔽通信提供了基础。利用以太坊的智能合约，我们可以在处理业务的同时实现区块链上的相互通信：发送方通过部署智能合约的方式将信息传递到区块链上，接收方可以通过以太坊浏览器 Etherscan[18]或以太坊钱包[19]进行查看。所以利用区块链进行信息隐藏不需要考虑通信渠道的问题。虽然这种隐写方案解决了鲁棒性和通信渠道问题，但仍要解决信息隐藏的抗检测性和隐藏容量问题。

因此本文提出了一套完整的基于区块链的信息隐藏框架。我们的框架以以太坊的智能合约为基础，通过合约源码或合约字节码嵌入秘密信息。根据 solidity 语言的语法规则，发送方可以通过替换某些等价的变量类型和等价的语句嵌入秘密信息，或者改变合约声明、函数声明的顺序嵌入信息，这种方案的隐藏容量至少可以达到 0.28%。编译 Solidity 源码可以生成以太坊虚拟机的字节码。发送方还可以通过重新排列字节码中的基本块的顺序嵌入秘密信息，这方案的隐藏容量达到 1.75%。

本文剩下的部分组织如下。第二部分介绍了本文的背景和相关工作。第三部分详细描述了基于区块链的信息隐藏框架，并且对每种隐写方法的原理做出了详细的解释。第四部分展示了实验的整体设计和实验结果。第五部分对全文做出了总结。

2 背景和相关工作

2.1 区块链技术

区块链是比特币的底层技术，首次出现在中本聪 2008 年发表的《比特币：一种点对点的电子现金系统》[20]中。文中详细描述了如何建立一套全新的、去中心化的、不需要信任基础的点到点交易体系的方法，其可行性已经被自 2009 年运行至今的比特币所证明。区块链技术解决了数字货币的两大问题：双重支付问题和拜占庭将军问题[21]-[26]。它被认为是继大型计算机、个人计算机、互联网、移动社交之后的第 5 次颠覆式计算范式，是人类信用进化史上继血亲信用、贵金属信用、央行纸币信用之后的第 4 个里程碑[27]。

区块链技术通过去中心化共识机制使网络中的每个节点都维护一份权威的、可信的、不可篡改的、公开的账本。利用 p2p 网络技术，区块链中的一笔交易会被广播到网络中的所有节点。其中一个完整的区块链客户端被称为全节点。每个全节点会根据综合标准对每笔交易和每个新区块进行独立验证。为了获得奖金和交易费，挖矿节点会竞争求解一道数学难题。难题的答案会被放入新区块中，作为矿工的工作量证明。在收到新的区块之前求解成功的矿工会广播新的区块。收到新的区块后矿工开始求解下一道难题。在这种工作量证明机制下，每个节点都会独立地选择累计工作量最大的区块链。通过每个节点之间的相互作用，最终整个区块链网络达成一种去中心化的共识机制。由于每个区块都有前一个区块的特征，想要篡改一个区块的交易记录，必须重新计算该块之后的所有区块。对于比特币网络，根据比特币网络算力以及现有计算设备综合考虑，一般一个区块后有六个区块就很难被篡改了。要想有效地对区块链网络进行攻击，攻击者必须具备全网 50%以上的算力。由于区块链网络的算力十分庞大，通过算力篡改区块已经成为理论上的攻击场景。

比特币是区块链 1.0 时代的代表。为了增加区块链应用的灵活性，区块链 2.0 向用户提供了可编程的脚本。其中，图灵完备的分布式智能合约系统以太坊是这个时代的代表。智能合约是部署在区块链上的去中心化、可信息共享的程序代码。签署合约的各参与方就合约内容达成一致，将内容以脚本的形式部署在区块链上，即可不依赖任何中心机构自动化地代表各签署方执行合约。智能合约具有自治、去中心化等特点，一旦启动就会自动运行，不需要任何合约签署方的干预。智能合约不仅赋予了区块链底层数据可编程性，为区块链 2.0 和区块链 3.0 奠定了基础；还封装了区块链网络中各节点的复杂行为，为建立基于区块链技术的上层应用提供方便的接口。智能合约具有广泛的应用场景。例如，对互联网金融的股权招募，智能合约可以记录每一笔融资，在成功达到特定融资额度后计算每个投资人的股权份额，或在超过某个时间界限仍未达到融资额

度时将资金退还给投资人。此外，智能合约还被广泛应用于房地产、医疗、汽车、供应链等行业。

由于智能合约应用广泛，每天都会有大量智能合约部署在区块链上。因此，我们可以利用智能合约代码的冗余空间嵌入秘密信息，当然也可以使用传统的文本信息隐藏算法嵌入信息。

2.2 相关工作

由于基于区块链的信息隐藏技术还没有被提出，我们在这里介绍一些跟本文相关的隐写和软件水印方面的相关工作。在[28]中，Rakan 等人提出了一种在 x86 二进制可执行文件中嵌入秘密信息的方案-Hydan，它的嵌入率可以达到 1/110。在[29]中，Bertrand 等人提出了一种评估二进制可执行文件隐写隐蔽性的框架，并且利用这个框架实现了因特尔的 IA-32 架构中的隐写。他们将嵌入率提高到 1/88.76，并且隐蔽性比 Hydan 要高。虽然这两种方案都同时考虑了隐蔽性和隐藏容量问题，但它们都没有考虑信息隐藏的抗干扰问题，所以它们的鲁棒性都比较低。微软公司在 1996 提出了一种利用重新排列基本块嵌入指纹的软件水印方案[30]。在这一技术刚公开时，它还是一个不完整的算法。随后，Myles 等人以及 Hattanda 和 Ichikawa 分别在参考文献[31]和[32]中进一步实现了细节。这种算法的优点是对程序的性能影响很小。然而它们很容易被击败-攻击者只要把程序中所有可以重排的东西重新排列一遍就可以破坏水印了。

在软件水印方面，最早有四个相关专利被发表[33]-[36]。这些方法往往比较简单，容易被攻击者发现和篡改。在[37]中软件水印技术被分为静态水印和动态水印。静态水印将程序看成一个静态对象，嵌入和提取水印不需要运行程序。针对静态水印，Moskowit 和 Cooperman 提出了一种新的防篡改方法[38]。攻击者随机篡改水印可能会导致程序无法运行。静态水印还有代码替换[39]、寄存器分配[40]、不透明谓词[41]等方案。虽然静态水印容易实现，但是在隐蔽性和防篡改性方面一般没有动态水印表现好。动态水印在程序运行的阶段嵌入和提取水印。Clark 提出了一种通过 Java 的映射机制来实现防篡改的方案[42]。该方案在程序运行过程中检验图节点是否被人为篡改，但它的隐蔽性较差，而且增加了程序的运行负担。Palsberg 提出了一种新的动态图水印防篡改方法[43]。该方案难以抵御语义保持转换攻击。He Yong 提出了一种新的基于常量编码的防篡改方案[44]。该方案有较高的隐蔽性，但难以抵御模式匹配攻击。动态水印还有 K 数编码[45]、排列编码[46]、PPCT 动态图编码[47]等方案。这些软件水印方案为了提高鲁棒性通常在已有的水印方案中加入防篡改算法，但这样做会使程序占用更大的空间、也会影响了程序的运行效率。

以上这些方案都有各自的优缺点，但由于载体媒介的局限性，它们很难在不影响程序的性能和大小的情况下达到较高的隐蔽性和鲁棒性。因此，本文提出了一种鲁棒性极强的基于区块链的隐写方案。由于我们的方案不需要加入防篡改算法，所以对嵌入算法的隐蔽性、程序的大小和运行效率影响较小。

3 基于智能合约的隐写术框架

3.1 系统框架

本文提出区块链信息隐藏框架如图 1 所示。其中秘密信息的发送方为 Alice，接收方为 Bob，假设有一个敌手 Jack 试图监听或者篡改 Alice 发送的消息。图中的 M′为秘密信息 M 经过 AES 加密后生成的秘密信息，Ccontract 为载体合约，Scontract 为含秘载体合约。首先 Alice 使用信息隐藏嵌入算法将 M′嵌入到载体合约中生成 Scontract，然后将 Scontract 部署到以太坊的区块链上。Bob 可以通过区块链获取到 Scontract，然后使用信息隐藏提取算法获取 M′。

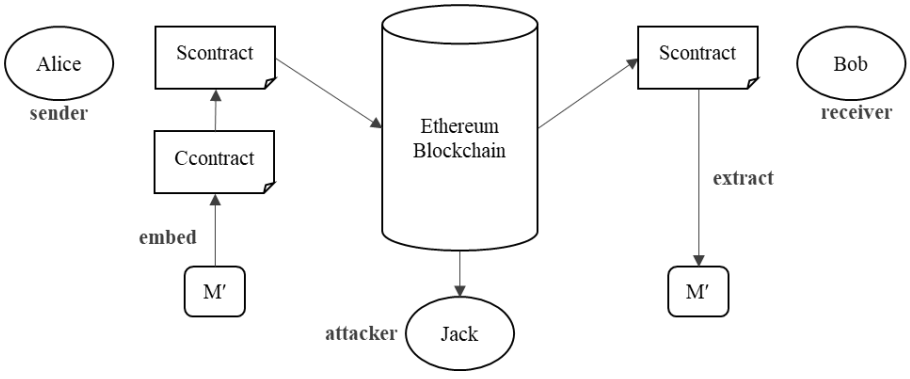

图 1　区块链信息隐藏框架

Alice 和 Bob 通信前共享密钥 K_0 、K_1。 其中 K_0 用于秘密信息 M 的加密和解密，K_1 用于信息隐藏嵌入和提取过程中的哈希加密。攻击者 Jack 可以查看以太坊区块链上的智能合约，但是他很难知道合约中是否隐藏信息，更无法篡改区块上的智能合约。

除了部署智能合约到区块链上，还有其他方法可以上传信息到区块链上。例如，利用以太坊的 web3.eth.sendTransaction() 方法，我们可以发送一个交易并在交易中放入信息的 16 进制码；我们还可以在一个智能合约函数中传递信息的 16 进制码，执行函数后信息的 16 进制码会被存储到区块链中。以上这两种方法都可以通过交易地址查看信息的 16 进制码。但由于监听者很容易知道发送方正在传递信息，而且一旦监听者将 16 进制码转化成 UTF-8 等编码就可以获得信息，所以这两种方法的隐蔽性都不高。即使是加密后的信息，监听者也可以花费一定的代价进行破解。我们提供的方案是在部署智能合约的阶段隐藏信息，而不是在执行智能合约函数（需要先部署智能合约）的阶段隐藏信息。这样做可以以智能合约为载体，在不影响智能合约功能的情况下进行隐蔽通信。监听者可以发现 Alice 部署了一个智能合约，但他不知道 Alice 是否正在传递消息。并且在区块链上部署智能合约是一件极其普通的事，不容易引起怀疑。即使监听者分析 Alice 发布的智能合约的源码或字节码，只要 Alice 使用的隐写方案的抗检测性够高，就不会被检测出包含秘密信息。

3.2 基于重新排序的隐写模型

在 Disappearing Cryptography[48]一书中，Peter Wayner 展示了一种通过修改无序列表中元素的排列顺序嵌入秘密信息的方案。也就是说如果某个对象中的某些元素的排列顺序不影响对象的使用，我们就可以通过改变这些元素的顺序嵌入秘密信息。在智能合约源码中，改变函数定义或声明的顺序不会影响合约的正常使用。在智能合约的字节码中，改变基本块的顺序也不会影响合约的使用。所以我们可以通过基于重新排序的隐写算法在智能合约中隐藏信息。

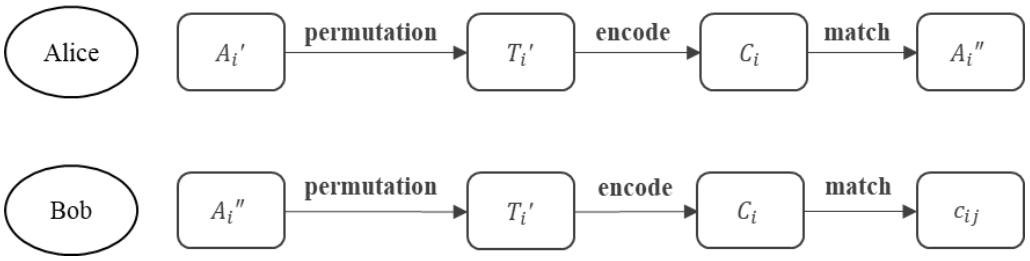

图 2 基于重新排序的隐写模型

图 2 描述了基于重新排序的隐写模型。其中，A_i'={a_1', a_2',…, a_n'} 为待排序的元素集合。Alice 先将 A_i' 进行全排列操作，并将排序后的结果进行字典排序，生成全排列集合 T_i'={A_{i1}', A_{i2}', … , $A_{in!}'$}，然后对 T_i' 的每个元素进行编码，生成编码集合 C_i ={C_{i1} , C_{i2} , … , $C_{in!}$}，接着匹配加密后秘密信息 M′ 和编码集合 C_i ，选择匹配到的编码 c_{ij} 对应的排列 A_{ij} 作为最终的排序结果 A_i'' 。Bob 收到排序后的集合 A_i'' 后再进行逆操作就可以提取出 A_i'' 对应的编码 c_{ij} 。算法 1 和算法 2 分别描述了基于重新排序的隐写算法的嵌入过程和提取过程。

算法 1：基于重新排序的隐写算法的嵌入过程

输入：M′，$A_i(a_1, a_2, \dots, a_n)$, K_1.

输出：$A_i''(a_1'', a_2'', \dots, a_n'')$.

步骤：

1) 初始化 i=1.
2) A_i'=H(A_i)={a_1', a_2', … , a_n'}={H(a_1+K_1), H(a_2+K_1), … , H(a_n+K_1)}.
3) $T_i'(A_{i1}', A_{i2}', \dots, A_{in!}')$=perm($A_i'$), 通过$T_i'$=H($T_i$)={H($A_{i1}$), H($A_{i2}$), … ,H($A_{in!}$)}确定$T_i$.
4) 根据编码规则，对$T_i(A_{i1}, A_{i2}, \dots, A_{in!})$中 n!个元素编码，得到编码集合$C_i(c_{i1}, c_{i2}, \dots, c_{in!})$.
5) 从最高位开始，匹配 M′中和$C_i(c_{i1}, c_{i2}, \dots, c_{in!})$中相同的编码。如果有相同的编码，如编码为$c_{ij}$(1<=j<=n!)，则$A_i''$=$A_{ij}$，输出$A_i''$，转到步骤 6); 如果没有相同的编码则在 M′后补 0,直到找到相同的编码c_{ij}，然后A_i''=A_{ij}，输出A_i''，嵌入完成，转到步骤 8).
6) M′ = M′ - c_{ij}，即从最高位开始去除 M′中的一串c_{ij}.

7) 如果 M′为空，则嵌入完成，转到步骤 8)，否则 i=i+1,返回步骤 2).

8) 对剩余的 A_i 进行排列，使其对应的编码都为 0.

算法 2：基于重新排序的隐写算法的提取过程

输入：$A_i''(a_1'', a_2'', \dots, a_n'')$，$K_1$.

输出：M′.

步骤：

1) 初始化 i=1.

2) A_i'=H(A_i'')={a_1', a_2', ... , a_n'}={H(a_1''+K_1), H(a_2''+K_1), … , H(a_n''+K_1)}.

3) $T_i'(A_{i1}', A_{i2}', \dots, A_{in!}')$=perm($A_i'$), 通过$T_i'$=H($T_i$)={H($A_{i1}$), H($A_{i2}$), … ,H($A_{in!}$)}确定$T_i$.

4) 根据编码规则，对$T_i(A_{i1}, A_{i2}, \dots, A_{in!})$中 n!个元素编码，得到编码集合$C_i(c_{i1}, c_{i2}, \dots, c_{in!})$.

5) 通过A_i''==A_{ij}，找到A_{ij}，从而获得c_{ij}.

6) M′=M′+c_{ij}，即将 M′与c_{ij}串联.

7) 如果 i==s,提取完毕，转步骤 8)，否则 i=i+1,返回步骤 2).

8) 去掉 M′末尾的所有 0，如果 M′的长度不为 128 的整数倍，在 M′的末尾补 0，直到 M′的长度为 128 的整数倍，输出 M′.

首先，Alice 使用密钥K_0对秘密信息 M 进行 AES 加密，并生成一串二进制比特流 M′。由 AES 加密的性质可知 M′的长度为 128 的倍数。需要排序的元素集合被定义为$A_i(a_1, a_2, \dots, a_n)$，其中 1<=i<=s，s 为总集合个数，$a_j$(1<=j<=n)为某个元素，n 为元素总个数。Alice 先使用密钥K_1计算每个元素和K_1串联后的哈希值 H(a_i+K_1)，记 H(a_i+K_1)为a_i'，得到$A_i'(a_1', a_2', \dots, a_n')$。并利用全排列函数 perm()对$A_i'$进行排列并对排列结果字典排序，得到的$T_i'(A_{i1}', A_{i2}', \dots, A_{in!}')$为$A_i'$的全排列的集合，且$T_i'(A_{i1}', A_{i2}', \dots, A_{in!}')$=H($T_i$)={ H($A_{i1}$), H($A_{i2}$), … ,H($A_{in}$)}。然后 Alice 对$T_i$进行编码，并根据 M′选择对应的排列$A_{ij}$，生成排列后的序列$A_i''(a_1'', a_2'', \dots, a_n'')$。

我们的编码规则参考了 Bertrand Anckaert 的论文[29]。如果 $\log_2 n!$ 为整数，$T_i(A_{i1}, A_{i2}, \dots, A_{in!})$对应的编码为每个$A_{ij}$(1<=j<=n!)对应的位置的二进制数（位置从 0 开始计算）。如果 $\log_2 n!$ 不为整数，则A_i最低能存储 $\lfloor \log_2 n! \rfloor$ 位二进制。为了不浪费剩余的 n!-$\lfloor \log_2 n! \rfloor$ 个排列，本文将剩余的 n!-$\lfloor \log_2 n! \rfloor$ 的编码与前 n!-$\lfloor \log_2 n! \rfloor$ 个编码对应，因此有(n!-2^$\lfloor \log_2 n! \rfloor$)/2^$\lfloor \log_2 n! \rfloor$的几率存储$\lfloor \log_2 n! \rfloor$ +1 位的二进制串。

排列：	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}
编码：	00(0)	01(0)	10	11	00(1)	01(1)

图 3 n=3 时的编码示意图

如图 3 所示，当 n=3 时，$T_1(A_{11}, A_{12}, A_{13}, A_{14}, A_{15}, A_{16})$对应的编码为$C_1$(000,010,10,11,001,011)，且有 1/2 的几率嵌入 3 位比特串。如果每个被嵌入的比特串有相同概率为 1 或 0，则可以嵌入的位数

$$b(n)= \lfloor \log_2 n! \rfloor + (n!-2^{\lfloor \log_2 n! \rfloor})/2^{\lfloor \log_2 n! \rfloor}。 \quad (1)$$

当 $\log_2 n!$ 为整数时，可以验证 b(n)=$\log_2 n!$ 。以这种方式编码可以最大限度的利用排列后的结果，从而提高算法的隐藏容量。

3.3 基于智能合约源码的隐写

基于智能合约源码的隐写模型如图 4 所示。其中 Scode 表示智能合约的源码，Bcode 表示智能合约的字节码。Alice 首先将加密后的秘密信息 M′嵌入到合约源码中，然后将源码编译生成字节码，接着部署字节码到区块链上。字节码部署成功后，Alice 可以将合约源码公开到 Etherscan 上。Etherscan 是以太坊的区块浏览器，利用它我们可以查看区块上的很多信息。Alice 公开合约源码时需要验证合约源码和区块上的字节码的一致性。公布成功后 Bob 就可以通过 Etherscan 查看合约源码，然后提取出 M′。

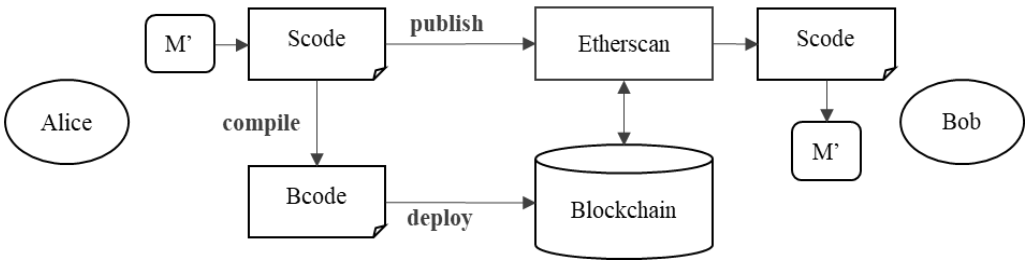

图 4 基于合约源码的隐写模型

以以太坊的 Solidity 语言为例，通过 Solidity 语言编写的智能合约源码具有 contract、library 和 interface 三种合约类型。这三种类型包含的函数定义和声明都可以任意改变位置而不影响程序的使用。我们以 contract 类型为例，我们可以将 contract 类型中的所有函数声明或定义视为 a_i(1<=i<=n，n 为函数定义或声明的总数)，一个 contract 视为 $A_i(a_1, a_2, \dots, a_n)$。因此，我们可以按照我们提出的基于重新排序的隐写算法重排 contract、library 和 interface 中的函数，生成一个包含 M′的含秘合约。发送方 Alice 可以将这个合约部署到区块链上，并且公布合约的源码。接收方 Bob 可以通过 Alice 的以太坊账户在以太坊浏览器上查看 Alice 部署的合约，然后通过基于重新排序的隐写算法提取 M′。

Solidity 语言中的某些类型具有不同的表达方式，例如 int/uint 分别表示 256 位的有符号/无符号整数，它们又分别和 int256/uint256 等价[49]。这种具有等价表达的参数还有 fixed、ufixed 和 byte 等。以 uint 为例，发送方 Alice 可以规定 uint 表示 0，uint256 表示 1。首先，嵌入程序需要遍历整个合约源码，找出所有的 uint 或者 uint256，然后跟据 M′依次修改匹配到的 uint 或者 uint256，如果 M′的某位为 0，则使用 uint，否则使用 uint256。二进制程序的信息隐藏通常会用到等价指令替换的方法，这是因为一条指令可以有多种不同的表达方式，这些不同的表达方式具有相同的功能。对于 Solidity 语言，同样存在许多可以替换的语句。例如，比较符号左右两边可以交换，a+=b 和 a=a+b 可以互换，String 类型的变量值可以使用单引号或双引号。因此我们也可以使用等价语句替换的方法隐藏信息。

Alice 公布合约源码时需要对合约源码进行验证，也就是验证公布的合约源码与上传到区块链上的合约字节码的一致性。合约源码是不会存储到区块链上的，只有合约源码编译生成的字节码会存储到区块链上。但是合约字节码的末尾会有 32 字节的 Swarm 哈希值，合约源码的一个空格的改变都会导致整个合约字节码的改变[50]。因此如果监听方 Jack 试图修改合约源码并公开源码，他在验证阶段会无法通过，因为他已经改动了合约源码，他又无法修改区块链中存储的智能合约字节码。如果 Alice 使用普通的网页信息隐藏，一旦 Jack 修改了网页中的某些内容（Jack 是名黑客），Bob 可能会获得错误的消息。在这个过程中，Jack 甚至不需要知道密钥和隐藏算法就可以破坏 Alice 和 Bob 之间的通信。所以跟普通的网页、文本等信息隐藏相比，以智能合约源码为载体的信息隐藏更加安全。

智能合约源码本质上是一段文本信息，所以传统的文本信息隐藏方法也可以在智能合约上使用。

3.4 基于智能合约字节码的隐写

基于智能合约字节码的隐写模型如图 5 所示。

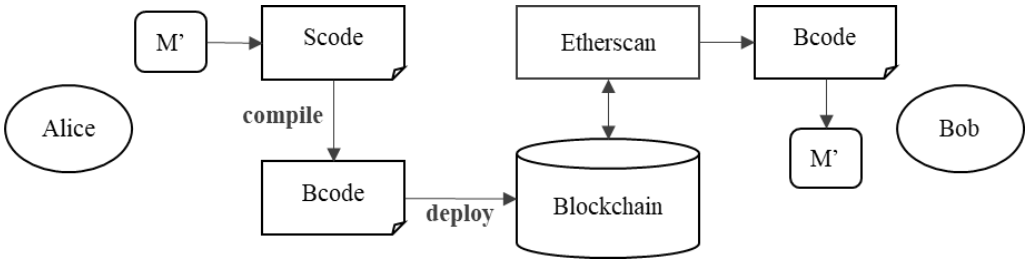

图 5 基于合约字节码的隐写模型

Alice 先将合约源码编译成字节码，然后在合约字节码中嵌入加密后的秘密信息 M′，接着将含秘字节码部署到区块链上。Bob 可以通过 Etherscan 或以太坊钱包查找到部署在区块链上的字节码。在这个过程中，Alice 不需要公布合约源码，所

以 Bob 只能查看到含秘字节码。因为攻击者 Jack 在不清楚合约源码的情况下很难分析出字节码是否包含秘密信息，所以基于合约字节码的隐写具有很高的隐蔽性。

基本块是指程序中一一顺序执行的语句序列，它只有一个入口和一个出口。在软件水印中，人们可以重新排列每个函数中的基本块嵌入水印信息。由于是通过重新排序的方法嵌入信息，我们可以使用 B 中提到的基于重新排序的隐写模型。假设一个函数中有 m 个基本块可以重排，那么这个函数可以隐藏 $\log_2 m!$ 位的信息。通过将合约源码编译成合约汇编码，我们可以看到汇编码中有许多类似“tag_数字”的标签。每个标签由若干个基本块组成。通过查看以太坊的黄皮书[51]，我们可以知道一个标签对应的助记符为 JUMPDEST,它标志着一个有效的跳转目的地，即这段程序的入口。以太坊通过 JUMP 或 JUMPI 指令完成基本块之间的跳转。JUMP 指令表示跳转到栈顶元素标志的地址。JUMPI 元素表示条件跳转，如果栈顶之后的元素不为 0 则跳转到栈顶元素标志的地址，否则顺序执行下面的指令。为了保持程序的语义不变，改变基本块的顺序同时需要改变某些跳转指令或者插入额外的跳转指令。

由于公开源码需要验证源码和字节码之间的一致性，而基于合约字节码的隐写修改了编译后的字节码，所以使用基于合约字节码的隐写不能公开合约源码。这也增加了合约字节码隐写的安全性，因为在不知道合约源码的情况下人们很难分析出字节码是否有过改动。此外，我们并不是直接在编译生成的字节码中嵌入信息，而是在 runtime 字节码中嵌入信息。这是因为编译生成的字节码包含两个部分，其中一部分用于智能合约的初始化，另一部分是合约的 runtime 字节码。如果合约源码没有被公开，人们无法通过 Etherscan 查看合约的初始化代码，只能看到 runtime 字节码。因此我们只在 runtime 字节码中进行隐写。

4 实验结果

4.1 实验准备

本文的实验中我们实现了一个完整的基于区块链的信息隐藏系统，包括预处理模块，秘密信息嵌入模块和秘密信息提取模块。系统的开发环境如下表所示：

表 1　系统开发环境

硬件环境	Intel i5-6500(3.2GHz)、8GB RAM
操作系统	Windows 10 64 位
编程语言	Java8
秘密信息文档类型	.sol
载体文档类型	.sol、.bin-runtime

我们首先在 Etherscan[52]上获取了 2 万份智能合约用于实验测试，并将它们保存为.sol 文件，然后使用 solc 命令将这些合约编译成.bin-runtime 文件。我们从网页上获取一些文本信息用于信息隐藏的实际测试。发送方使用 web3j 将一份含秘合约部署到 Rinkeby 测试链上。接收方通过查询部署合约的以太坊钱包地址可以获取含秘合约，然后通过提取算法提取出秘密信息。

4.2 隐藏容量分析

信息隐藏中的最大嵌入率指的是最大能嵌入的秘密信息的比特数与载体的比特数的比值。我们提出的各种方法的嵌入率如表 2 所示。

表 2 平均嵌入率

隐藏方法	说明	平均嵌入率
声明排序	对函数声明、变量声明的排序	0.14%
变量类型替换	例如 uint 和 uint256 的等价替换	0.06%
源代码等价语句替换	例如 a+=b 和 a=a+b 的等价替换	0.08%
基本块重排	重排字节码中的基本块	1.75%

我们对 2 万份合约进行了测试，并计算出每种隐写方案在 2 万分合约中的平均嵌入率。其中，基于智能合约源码的隐写的嵌入率至少为 0.28%，基于智能合约字节码的嵌入率为 1.75%。基于合约源码的隐写只考虑了本文提出的几个方法，并没有加入一些传统的文本信息隐藏方法，所以隐藏容量较低。而基于智能合约字节码的最大嵌入率高于[4]中提出的方法，而且我们提出的方法有更高的鲁棒性和抗检测性。

4.3 鲁棒性分析

鲁棒性是智能合约隐写的一个重要优点。基于合约源码的隐写的鲁棒性要远远高于普通的隐写术。由于公开合约源码需要验证合约源码和部署到区块链上的字节码的一致性，所以被篡改的合约源码无法进行公开。即使篡改者攻破了 Etherscan 的验证机制，而且成功公开了篡改后的合约源码，接收方也可以使用编译器轻松检测到合约源码是否被篡改。因为一旦合约源码被篡改，它编译生成的字节码和区块链上的字节码将会不同。而基于合约字节码的隐写的鲁棒性完全取决于区块链的防篡改性，显然它的鲁棒性更高于其他的隐写术。因为篡改者的算力至少要占全网算力的 50%才有可能完成篡改，所以已经部署到区块链上的合约字节码几乎是不可能被篡改的。

4.4 安全性分析

本文提出的基于智能合约的隐写术的安全性包括 AES 加密算法的安全性、带密钥的 MD5 算法的安全性、以太坊区块链系统的安全性。假如攻击者察觉了合约中包含秘密信息，他需要破解带密钥的 MD5 算法并且了解嵌入、提取算法才能获得加密后的秘密信息。然后，攻击者破解 AES 算法才能获取秘密信息。如果攻击者想篡改秘密信息，还必须攻破以太坊的区块链系统。因此，本信息隐藏系统有很高的安全性。

4.5 性能过载分析

性能过载分析指的是分析嵌入信息对程序性能的影响，它分为空间过载和时间过载。空间过载指的嵌入信息后程序的空间变化，时间过载指的是嵌入信息后程序的执行时间变化。虽然嵌入秘密信息后的智能合约仍可以正常部署使用，但我们仍需分析嵌入秘密信息对合约占用空间和部署速度的影响。对含秘合约的性能过载分析结果如表 3 所示。

表 3 性能过载分析

合约	平均占用空间	平均部署时间
原始载体合约	7013 bytes	13.467 ms
含秘载体合约	6983 bytes	13.482 ms

我们对 2000 份智能合约进行了测试。从表中可以看出含秘载体合约的占用空间略低于原始载体合约。这是由于等价指令的占用空间相近，所以等价替换嵌入算法对合约的空间影响较小。而且排序嵌入算法基本不改变合约的大小，所以总体上我们的方案对合约的空间影响很小。从表中可以看出含秘载体合约的平均部署时间略高于原始载体合约，差值为0.015 毫秒，这几乎可以忽略不计。这是因为虽然我们改变了合约中一些函数或语句的位置、一些语句的表达方式或是一些基本块的位置，但这些改变远没有添加额外的嵌入程序或是防篡改算法对程序的性能影响大。

5 总结

虽然信息隐藏技术已经趋于成熟，但基于区块链的信息隐藏至今还未被提出。所以我们的方案为信息隐藏技术提供了一个新的研究方向。基于区块链的信息隐藏可以将区块链的不可篡改的优点完美的应用到信息隐藏当中。区块链的不可篡改性也就转变成了信息隐藏的鲁棒性，也就是说传统的语义保护转换、代码混淆等攻击方式都将失效。即使攻击者不再局限于获取秘密信息，而是单纯的破坏秘密信息，只要区块链技术没有被攻破，攻击者都将无功而返。基于智能合约的隐写术在没有改变智能合约基本特征的情况下隐藏信息本身就具有较高的抗检测性。基于合约字节码的信息隐藏不公开合约源码，具有更高的抗检测性。目前我们已经发现的某些隐写算法的隐藏容量和传统的程序信息隐藏容量相近，但我们的方案有更高的鲁棒性，而且提供区块链网络这种较安全的通信渠道。根据我们提出的区块链信息隐藏框架，仍然有一些隐藏方案未被发现。今后，我们将继续寻找更优的隐藏方案，提高隐藏容量。我们的实验还可以继续完善。今后，我们将对智能合约隐写术进行隐写分析，然后跟据隐写分析的结果对隐写方案进行改进。

References:

[1] M. Walport. Distributed ledger technology: beyond block chain. U.K. Government Office Sci., London, U.K., Tech. Rep., Jan. 2016.

[2] W. Bender, D. Gruhl, N. Morimoto, A. Lu. Techniques for data hiding. IBM Syst, J., vol. 35, no. 3–4, 1996, pp. 313–336.

[3] F. A. P. Petitcolas, R. J. Anderson, M. G. Kuhn. Information hiding—A survey. Proc. IEEE, vol. 87, no. 7, Jul.1999, pp. 1062–1078.

[4] M. Wu, H. Yu, A. Gelman. Multi-level data hiding for digital image and video. SPIE Photonics East, Boston, MA, 1999.

[5] N. F. Johnson, Z. Duric, S. Jajodia. Information Hiding: Steganography and Watermarking—Attacks and Countermeasures. Norwell,MA: Kluwer, 2001.

[6] R. Chandramouli, M. Kharrazi, N. Memon. Image steganography and steganalysis concepts and practice. DigitalWatermarking Lecture Notes in Computer Science 2939, 2004, pp. 35–49.

[7] P. Tsai, Y. C. Hu, H. L. Yeh. Reversible image hiding scheme using predictive coding and histogram shifting. Signal Process, vol.89, 2009, pp. 1129–1143.

[8] T. Pevný, T. Filler, P. Bas. Using high-dimensional image models to perform highly undetectable steganography. Information Hiding(Lecture Not es in Computer Science), vol. 6387. Berlin, Germany:Springer-Verlag, 2010, pp. 161–177.

[9] J. Xu et al. Hidden message in a deformation-based texture. Vis. Comput. Int. J. Comput. Graph, vol. 31, no. 12, 2015, pp. 1653–1669.

[10] A.Wilson, P. Blunsom, A. D. Ker. Linguistic steganography on twitter: hierarchical language modeling with manual interaction. in IS&T/SPIE Electronic Imaging, 2014, pp. 201–217.

[11] Q.J. Zhao, H.T. Lu, X.H. Jiang. Web page watermarking for tamper-proof. Journal of Shanghai Jiaotong University, vol. 3, 2005, pp.280-284.

[12] Q.J. Zhao, H.T. Lu, X.H. Jiang. A PCAbased watermarking scheme for tamper-proof of web pages. Journal of Pattern Recognition, Vol.38, 20 05, pp.1321-1324.

[13] Ethereum white paper. A next-generation smart contract and decentralized application platform[EB/OL]. https://github.com/ethereum/wiki/wiki/Wh ite Paper

[14] Augur. http://www.augur.net/.

[15] Skuchain. http://www.skuchain.com/.

[16] http://koinify.com.

[17] A. K. R. Dermody and O. Slama. Counterparty announcement. https://bitcointalk.org/index.php?topic=395761.0

[18] https://etherscan.io/

[19] https://github.com/ethereum/mist/releases/

[20] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http://bitcoin.org/bitcoin.pdf, 2009.

[21] The Byzantine Generals problem[EB/OL]. http://research.microsoft.com/en-us/um/people/lamport/pubs/byz.pdf.

[22] FAN J, YI L T, SHU J W. Research on the technologies of Byzantine system[J]. Journal of Software, 2013, 24(6):1346-1360

[23] NELSON M. The Byzantine General's problem: an agreement protocol for distributed system[EB/OL]. http://www.drdobbs.com/cpp/the-byzantine -generals-problem/206904396.

[24] LAMPORT L. The weak byzantine generals problem[J]. Journal of the ACM (JACM), 1983, 30(3): 668-676.

[25] FEDOTOVA N, VELTRI L. Byzantine generals problem in the light of P2P computing. The International Conference on Mobile & UbiquitosS ystems: Networking & Services, 2006:1-5.

[26] REISCHUK R. A new solution for the byzantine generals problem[J]. Decision Support Systems, 1985, 1(2):182.

[27] SWAN M. Blockchain: blueprint for a new economy[M]. USA: O'Reilly Media Inc, 2015.

[28] El-Khalil, R., Keromytis, A.: Hydan: Hiding information in program binaries. In International Conference on Information and Communications Security, LNCS, Volume 3269, 2004.

[29] Bertrand Anckaert, Bjorn De Sutter, Dominique Chanet, and Koen De Bosschere. Steganography for executables and code transformation signa tures. In P.Choosnik and C.Secongtaek, editors, Information Security And Cryptology-ICISC 2004, number 3506, pages 425-439, Germany, April 20 05.

[30] Robert L, Dacidson and Nathan Myhrvold. Method and system for generating and auditing a signature for a computer program. U.S. patent 5 559884, September 1996. Assignee: Microsft Corporation.

[31] Kazuhiro Hattanda and Shuichi Ichikawa. The evaluation of Davidson's digital signature scheme. IEICE Transactions, 87-A(1):224-225,2004.

[32] Ginger Myles, Christian Collberg, Zachary Heidepriem, and Armand Navabi. The evaluation of two software watermarking algorithms. Softwar e:Practice and Experience, 2005, 35(10): 923-938.

[33] Samson, P.R.: Apparatus and method for serializing and validating copies of computer software. US Patent 5, 287, 408, 1994.

[34] Davidson, R.L., Myhrvold, N. Mehod and system for generating and auditing a signature for a computer program. US Patent 5,559,884,1996.

[35] Moskowitz, S., Cooperman, M. Method for stega-cipher protection of computer code. US Patent 5, 745, 569, 1996.

[36] Holmes, K. Computer software protection. US Patent 5, 287, 407, 1994.

[37] Collberg, C., Thomborson, C. On the Limits of Software Watermarking. Technical Report 164. Department of Computer Science, The Universi ty of Auckland, 1998.

[38] Moskowitz SA, Cooperman M. Method for stega-cipher protection of computer code[P]. American Pat 5, 745, 569, 1996.

[39] C. Collberg and T. R. Sahoo. Software watermarking in the frequency domain: Implementation, analysis, and attacks. J. Comput. Secur., vol. 13,no. 5, 2005, pp. 721-755.

[40] H. Lee and K. Kaneko. Two new algorithms for software watermarking by register allocation and their empirical evaluation. in Proc. IEEE In t. Conf. Inf. Technol., Apr. 2009, pp. 217-222.

[41] G. Arboit. A method for watermarking java programs via opaque predicates. in Proc. Int. Conf. Electron. Commerce Res., 2002, pp. 102-110.

[42] Jasvir Nagra, Clark Thomborson. Threading Software Watermarking. In:Proc 6th International workshop on Information Hiding(IH 2004), INCS 3200, Springer-Verlag. 2004, 208-333.

[43] Jen Palsberg, Sowmya Krishnaswamy, Minseok Kwon, Di Ma, Qiu yuan Shao, Yi Zhang. Experience with Software Watermarking[C]. In:Epst ein J,et al,eds.Pro.of the 16th Annual Computer Security Applications Conferences(ACSAC 2000). New Orleans:IEEE Computer Society Press,2000:3 08-316.

[44] Yong He. Tamper-proofing a Software Watermark Encoding Constants[D]. Master's thesis, Comp. sci. Dept. Univ. of Auckland, 2002

[45] W. Zhu, C. Thomborson, and F.-Y. Wang. A survey of software watermarking, in Proc. Int. Conf. Intell. Secur. Inform., 2005, pp. 454-458.

[46] M. Chroni and S. D. Nikolopoulos. An embedding graph-based model for software watermarking. in Proc. Int. Conf. Intell. Inf. Hiding Multi media Signal Process., 2012, pp. 261-264.

[47] J. Chen, S. Dai, and J. Chen. An improved software watermarking scheme based on PPCT encoding. in Proc. Int. Symp. Comput. Intell. Desi gn, 2016, pp. 341-344.

[48] Perter Wayner. Disappearing Cryptography: Information Hiding: Steganography and Watermarking (2nd Edition). Morgan Kaufamnn Publisher I nc., San Francisco, CA, 2002.

[49] https://solidity.readthedocs.io/en/develop/types.html

[50] https://solidity.readthedocs.io/en/develop/metadata.html

[51] DR. Gavin Wood. Ethereum a secure decentralised generalised transaction ledger, Byzantium version e94ebda, 2018-06-05 pp.32.

[52] https://etherscan.io/contractsVerified