盖吉斯之戒：窥探恶意智能合同的未来

Ari Juels
Cornell Tech, Jacobs Institute
IC3†
juels@cornell.edu

Ahmed Kosba
University of Maryland
akosba@cs.umd.edu

Elaine Shi
Cornell University
IC3†
rs2358@cornell.edu

摘要

由于具有可匿名（或可以使用假名）的特点以及不需要信用中介的优越性，类似比特币这样的加密货币已经在商业至于社会的各个领域创造了极大的增长。不过，这些增长有些也存在于犯罪领域，比如洗钱、黑市，以及各类勒索软件。

下一代加密货币（如以太币）将在一种自主成为交易媒介的程序——智能合同的支持下拥有更加丰富的脚本语言。在后文中，我们将探究智能合同推动犯罪生态系统发展的潜在危险性。更具体地说，我们将会剖析恶意智能合同（criminal smart contracts,CSCs）如何促进机密信息的泄露、密钥的失窃，以及引发现实世界中的各种犯罪（谋杀、纵火、恐怖袭击等）。

我们发现，以泄露信息为目的（按维基解密方式）的恶意智能合同在现存脚本语言中可行并且效率很高。有些恶意智能合同甚至可以使用简易的工具，比如 SNARKs（Succinct Non-interactive ARguments of Knowledge 简明非交互协议），来达到盗窃密钥的目的。SNARKs 目前已经可以在上述脚本语言中进行表述，并且预期将有更大范围的支持语言。同时，我们也发现智能合同正在慢慢显露的另一个特点——真实有效的数据反馈也能促进恶意智能合同的现实犯罪。

我们的研究结果着重指出：实行针对性政策并且发明保护性技术来抵制恶意智能合同已经成为当务之急。智能合同应当像它在被发明时承诺的那样——成为推动社会进步的工具。

关键词

恶意智能合同（CSCs）、以太币

盖吉斯之戒是在哲学家柏拉图的第二本书中提到的神话性质的石器。持有它的人能够随意隐身。——维基百科 • 盖吉斯之戒

CCS’16, October 24-28, 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2976749.2978362

1、 简介

类似比特币的加密货币从基础货币交易中移除了第三方，并且提供点对点的匿名（假名）交易平台。虽然这一特点有很大的应用空间，但也可能带来恶果。比特币就已经刺激了一大批勒索软件【6】、洗钱行为【38】以及非法商业活动的增长，正如臭名昭著的“丝绸之路”列举的一般。【31】

类似以太币的加密货币，以及类似 Counterparty【45】和 SmartContract【1】的系统能够提供比比特币更加丰富的功能。它们支持一种被记入图灵完备脚本语言的表示程序遗传项——智能合同。在一种类似以太坊的完全分布式系统中，智能合同使得普遍平等交换（原子互换）能够在无第三方的情况下进行，因此能够有效保证数据以及服务传输的成功及报酬支付。在这种智能合同系统具有很强灵活性的条件下，我们可以预见到这个系统能催发的不仅仅是有益的服务，还有犯罪的新形式。

我们在这里指的是能在分布式智能合同系统中促进犯罪的智能合同，比如 CSCs。举个简单的例子，以私人密钥盗窃为目的的就是一个 CSC。像这样的 CSC 可能需要为制定密钥 sk（比如认证机构的私人数字签名密钥）的机密传输支付一定费用。

我们在本论文中探讨以下问题：相比早期加密货币（比特币），CSCs 是否会提供范围更大的犯罪机会？这些犯罪的实现可能有多大？和传统的网络系统相比，CSCs 为犯罪提供的关键性优势是什么？探

究这些问题对发现威胁、设计对策有很大的意义。

1.1 CSC 遇到的挑战

将行犯罪在 CSCs 的构建中主要面对两个问题。第一，该 CSC 对于一种特定犯罪（比如密钥盗窃）是否完全可行是无法得到即时结论的。这是因为很难确保 CSC 实现了一个关键属性，在本文中我们称其为佣金公平（commission-fair），通俗来讲意思是它的执行要么保证了犯罪实施和相称的犯罪者收益，要么两方皆空。平等交易对于保证佣金公平来说是必要不充分条件：我们将展示在 CSC 构建实施平等交易时仍然有 CSC 作假的可能性。正确的 CSC 结构也因此变得脆弱不堪。

第二，即使 CSC 在理论上能够构建，在现行智能合同系统的有限操作码条件下，CSC 能否被实际创造出来还尚未明确。CSC 能够在避免过分繁重的计算工作的情况下执行，在有些智能合同系统中这也意味着极高的执行费。

下面举一些例子说明这些问题。

EXAMPLE 1A （密钥和解合同）：承包人 C 要求盗窃并传输受害者证书颁发机构（certificate authority,CA）CertoMart 的签名密钥 skV。C 向犯罪者 P 提供了一份报酬$reward 让他将 CertoMart 的密钥 skV 传输给 C。

为了保证密钥和报酬在比特币系统中的公平交易，C 和 P 需要一个具有信用的第三方机构或者直接沟通，提出被对方欺骗或者被法律效力发现的危险性。他们可以使用声誉系统相互审查，但这样的系统经常被执法部门【55】发现。相反，分散式的智能合同系统可以实现自我强制的平等交易。对于密钥盗窃，以下例子中使用 CSC Key-Theft 是可能的：

EXAMPLE 1B（密钥和解 CSC）：C 生成一个私钥/公钥对（skC，pkC），并用公钥 pkC 和 pkV（CertoMart 公钥）初始化 Key-Theft。Key-Theft 等待犯罪者 P 输入一对（）ct，π，π是零知识证明 ct = encpkC[skV] 是良好的。然后 Key-Theft 证明π并向 P 发送成功的报酬$reward。然后承包人 C 可以下载并解密 ct 获取和解密钥 skV。

Key-Theft 在 C 和 P 之间实行了平等交易——当且仅当 P 传输了有效密钥（被π证明），P 才能收到报酬，这样排除了对信用第三方的需要。但这仍然不是佣金公平，因为他不嗯能够保证 skVICT 拥有实际价值。CertoMart 可以抢先撤销自己的证书，然后认领 C 的报酬$reward。

正如我们已经提到的，本文的一个主要推动力就是探究犯罪者们是如何克服这些问题，成功地构建佣金平等的 CSCs。（对密钥协议来说，有必要使合同能够取消。）另外，我们发现 CSCs 可以使用已有的加密货币工具或者为加密货币展望的特点来有效实现（比如 zk-SNARKS【22】）。

1.2 关于本文

我们发现可能在智能合同系统中构建佣金公平的 CSCs 犯罪类型有三种：

1、机密文件的泄露或贩卖；

2、私钥盗窃；

3、“Calling Card 犯罪”——广泛类别的现实世界犯罪（谋杀、纵火等）。

CSCs 理论上可行，这个事实并不令人感到惊讶。但是在以前，我们还不清楚 CSCs 的现实性和广泛使用的可能到底有多大。正如佣金公平的 CSCs 构造显示，构建 CSCs 并非看起来这么直接，但是新的加密技术以及智能合同设计新方案能够使得他们可行。而且，毫无疑问犯罪者们将从超过本文以及整个社会能预测到范围之外设计 CSCs。

因此我们的工作展示了社会必须思考要抵御 CSCs 的方法。匿名引导下的犯罪活动已经对比特币的应用构成了主要障碍。在公共论坛上关于加密货币【16】恶意合同以及 2015 年 7 月成立以太坊的讨论还寥寥无几。只有在 CSCs 生命的早期认清它的危害，社会才能发展出及时的对策去控制它们，并且见证分布式智能合同系统完全实现。

我们的关注点是防范邪恶，不过令人高兴的是我们提出的技术也可以用来创造有益的合同。我们研究组织 CSCs 的技术和最前沿的加密工具（如 SNARKs）的使用。就像有益的智能合同的设计一样，CSC 构造要求密码学与佣金公平的融合。【34】

总之，我们的结论有：

- **恶意智能合同**：我们受下一代加密货币中的图灵完备脚本语言启发，发起 CSCs 的研究。我们探究三种类型的 CSCs 犯罪：机密信息泄露（第四节），密钥协议/盗窃（第五节），以及“Calling Card 犯罪”（比如暗杀）使用认证数据源（第六节）。我们探究了这些恶意合同引发的挑战，并且解释（展望）了新的技术以抵制中立并且实现佣金公平。我们一再强调这一点，因为佣金公平意味着承包方获得他们期望的实用性，明确的度量。佣金公平必须在明确给定 CSC 的情况下被定义。因此我们在网上的完整版本中正式指出了我们每一个 CSC 构造的佣金公平。【42】

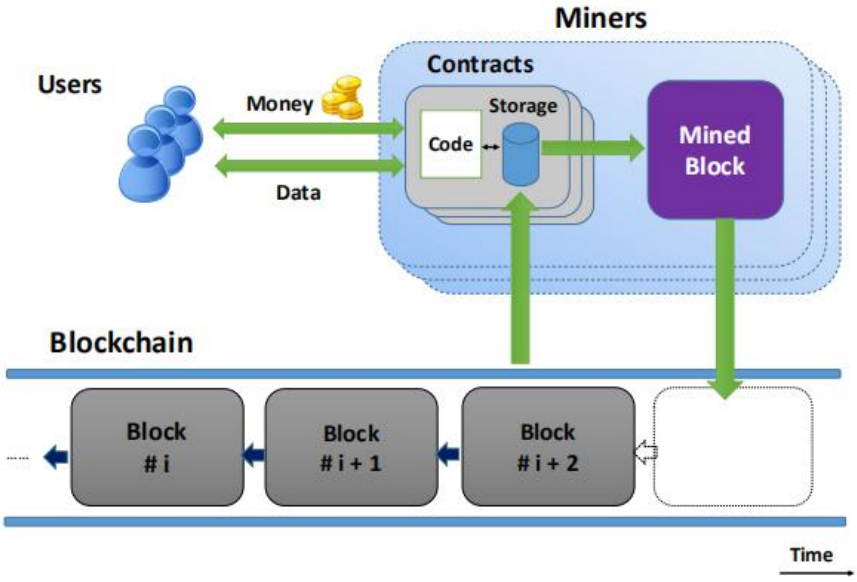

图 1：携带智能合同的分散式加密货币系统，由 Delmolino 等人完成。【34】智能合同的状态储存在公共区块链上。智能合同程序由矿工网络执行，他们在执行输出结果上达成共识，并且据此更新区块链上的合同状态。用户可以向合同发送钱或数据，或者从合同中收取。

• **概念证明**：为了说明即使是十分复杂的 CSC 也是现实的，我们将报告（在它们各自的部分）我们调查的 CSCs 实施方法。以泄露机密信息为目的的 CSC 在现行智能合同语言中能够高效实现，而以密钥盗窃或是“Calling Card 犯罪”为目的的 CSC 则分别依赖效率以及加密货币社区所设想的特点及可实现性。

• **对策**：我们在第七节中简要讨论了本文中提到的工作是如何帮助抑制 CSCs 的增长的。简而言之，为了达到最高的效率，CSCs 必须被展示出来，好让他们能带给社会警示。矿工们的经济激励不包括块状 CSC 交易，因为 CSCs 降低了市场中加密货币的价值。因此，意识和和强大的检测战略可以提供有效的一般性防御。我们工作的关键点事展现对这种对策的需求，以及激发在智能合同系统中实施对策的探索。

我们还在网上完全版本【42】中简要讨论了如何使类似硬件信任根源（比如 Intel SGX【40】）的技术更加成熟，以及程序困惑如何增长可能的 CSCs 的空间，或是使智能合同受益。

2、背景与相关工作

正在兴起的分散式加密货币【52，59】是以新兴区块链技术为根基的。在区块链技术中，矿工不仅要就数据达成共识，也要就计算达成共识。不严格地说，比特币-区块链系统核实了交易并且建立贮存了一个全球分类账本，这个账本被视为公共记忆的一部分，它的可靠性来源于底层分布式共识协议的执行。比特币支持一定范围的由区块链执行的可编程逻辑。不过，根据以往对于建造在比特币之上的智能合同类的应用得出的经验，它的脚本语言是限制性的，而且很难使用。【23,17,7,53,46】

当区块链的计算推广到任意图灵完备的逻辑，我们可以得到一个更加强力，更具普遍意义的智能合同系统。这种分散式智能合同系统的第一个体现是最近创立的以太币【59】。简单来说，在这样一个系统中，一个智能合同可以被看作代码的自主执行部分，它的输入以及输出都包括钱。（我们将在后文中给出更正式的定义。）相关的爱好者以及企业已经开始在以太币的基础上建造以及分叉系统，以发展各种各样的智能合同应用——类似证券和衍生品交易【45】、市场预测【5】、供应链源头【11】以及群众集资【2,47】。

图 1 显示了一个智能合同系统的高级结构，实例化自分散式加密货币。假设只要加密货币使用的基础共识协议是安全的，大部分矿工（由计算得）就能正确执行合同的可编程逻辑。

分散式智能合同系统的实例依靠 gas 来保护矿工以免受到拒绝服务攻击（比如运行死循环合同）。

Gas 是一种交易费用的形式，简略地说和合同运行时间相称。

在本文中，尽管我们没有在智能合同标记法中明确地表达 gas，但是我们尝试让程序逻辑尽可能地远离合同，为了保证 gas 以及交易费用够低，这是一种最佳选择。比如，我们提出的部分合同包括用户端执行的程序逻辑，在安全性上没有损失。

2.1 智能合同的优缺点

分散式智能合同有许多有益用途，比如实现多种新型金融工具。正如比特币对于交易的意义，在一个分散式智能合同系统内，共识系统强制合同自动执行。没有一个实体或者一部分实体能够干预这个过程。由于合同都是自我强迫执行的，它们排除了第三方信用机构或是声誉系统（原本用于降低交易危险性）。分散式智能合同相对于传统加密货币有以下优势：

• 在相互不信任，但配备编程可表达逻辑合同规则的两方之间实现平等交易。这个特点防止任意一方被另一方利用终止交易协议进行欺诈，也排除了对物理集结地和第三方中介（潜在性欺诈）的需要。

• 最小化各方之间的交互，减少不必要的监视和追踪的机会。

• 充实交易的外部状态，允许输入由进行物理或其他智能合同系统之外的活动中间人提供的认证数据，比如股票行情、天气预报等。这些在以太币中都处于起步阶段，但是它们的可用性在提高，使用可信硬件承诺可以刺激他们的调度。【61】

不过，对于所有的优势，同样也存在各种弊端，这些弊端会悄然促进犯罪发生：

• 平等交易使得相互不信任的犯罪方之间的交易得以完成，排除了对现今脆弱的声誉系统以及有潜在欺诈可能或执法渗透的第三方中介的需要。【55,39】

• 最小化互动导致执法活动更难进行。在有些情况下，对于我们先前提出的密钥盗窃以及“Calling Card 犯罪”CSCs 来说，罪犯可以建立一个合同然后逃跑，在没有进一步互动的情况下让合同自动执行。

• 具有丰富外部状态的交易拓宽了可能出现的 CSCs 范围，比如各种现实犯罪（恐怖主义、纵火、谋杀等）。

由于分散式智能合同系统通常继承了比特币的匿名（假名）特点，它们也会为犯罪活动提供相似的保密能力。因此，分散式智能合同系统的能力将会使新的地下生态系统以及社区成为可能。

2.2 数字货币与犯罪

比特币与智能合同不代表加密货币的最早现身。匿名电子现金在 1982 年 David Chaum 的研讨论文中被提出。【29】Naccache 和 Non Solms 指出，匿名货币会帮助“完美犯罪”，比如绑架这样难以被执法部门追踪的犯罪。【57】这一发现促使了公平盲签名的设计以及电子现金的“代管”，【26,58】使信用第三方连接身份认证与支付。这样的连接在经典电子现金方案中是可能的，用户可以在匿名取出现金时认证自己的身份，但是这不能应用于类似比特币的假名加密货币。

勒索软件自从 1989 年起就已经出现了【18】。由于勒索软件 CryptoLocker 据说已经收入了数亿美元的赎金【25】，加密病毒学【60】对勒索软件的一个重要改进已经被运用于比特币【44】。在 1995-6 的一篇题为《政治暗杀》的文章中使用匿名数字现金的暗杀市场被首次提出。【19】

社会各界已经对比特币犯罪进行了广泛的研究，比如洗钱【51】、比特币盗窃【49】，以及没法市场（如“丝绸之路”【31】）。Meiklejohn 等人【49】指出，比特币使用假名，被设计出来实现比特币匿名功能的机械不操控大量货币，通常来说，犯罪很难匿名取出大量现金。

另一方面，Ron 和 Shamir 提供的证据表明，FBI 未能找到最多比特币持有者 Dread Pirate Roberts（Ross Ulbricht）——“丝绸之路”的操控者——即使是在找到他的笔记本电脑之后【56】。Möser, Böhome 和 Breuker【51】发现他们无法成功使研究中两种混合或三种混合的交易中让被隐去的信息还原。这说明“Know-Your-Customer”原则——监管机构反洗钱的主要工具——可能在加密货币领域难以奏效。日趋实际的提案提出用 NIZK 证明加密货币中的匿名性【20,33,50】，而且至少有一种在商业部署的早期阶段承诺给出对犯罪有效的更强的匿名性。【13】

3、记号法和威胁模型

我们采用了 Kosba 等人提出的正式区块链模型。【43】作为背景，我们给出了这一部分中模型的高级表述。在论文中，我们用这个模型详细说明了加密货币协议；这些协议包括 CSCs 和相应的用户侧协议。

智能合同模型中的协议：我们的模型将合同视为特殊方，它被委托行使正确性而非私密性，正如上文指出的那样。（当然在现实中，这个合同是由网络操

控的。）所有发送给合同的信息和它的内状态都是公开的。这个合同利用信息交换与用户和其他合同交互（或指交易）。用账户余额形式表达的钱被记录在全球性分类账本上（区块链上）。合同可以连接账本并且更新数据来实现钱在用户之间的转账，而他们均由假名公钥代表。

3.1 威胁模型

我们在本文中采用以下威胁模型。

• 区块链：信任正确性而非隐私。我们假设区块链总是能够正确储存数据并进行计算，而且总是有效。区块链向公众展示它所有的内部数据，没有私有的数据。

• 任意恶意契约方：我们假设合同各方并不相互信任，他们的行为只是为了自身利益最大化。他们不仅可以从规定协议中随意脱离，而且还能贸然终止协议。

• 竞争对手的网络影响：我们假设区块链间的信息和用户在一定限度的延迟内传递。（一个用户总是可以执行由恶意矿工掉落的一笔交易。）在我们的模型中，一个竞争对手会立即接收到并且可以随意记录信息。在现实的分散式加密货币中，胜利的矿工可以设置信息处理的命令。而对手可能会交结部分矿工或者在节点间影响信息传播。就像我们会在第五节中提到的，对于密钥盗窃合同来说，信息记录使一个佣金公平 CSC 必须避免的冲击性攻击成为现实。

我们采用的正式模型（在本节后段会回顾并完整表述【43】）抓住了威胁模型的各个方面。

3.2 安全定义

CSC 要做到佣金公平有两个要求：

• 佣金公平的正确定义。佣金公平没有通用的正式定义：它特定于应用程序，取决于犯罪行为（和犯罪者）的目标。因此，对于每一个 CSC 来说，我们在网上完整版本【42】中指定一种佣金公平的相对定义，借助于 UC 风格的理想型功能来实现它。仅仅指出正确的理想型功能已经是一个巨大的挑战了！我们将在第五节解释这个挑战。使用最朴素密钥的网上完整版本代表着看上去正确，实际上有缺陷的密钥盗窃合同。

• 正确的协议实现。为了证明 CSC 是佣金公平的，我们必须说明它的（现实）协议能够模拟相应的理想型功能。我们证明这一点是为了说明标准通用可组合（UC）激励样式中的 CSCs，在密码学文献中被采用的，反对任意恶意合同交易对手以及可能的网络对手。我们的协议也很安全，可以防止对手中止协议，比如尝试在还未支付给另一方费用时中止协议。在中止行为存在的情况下，公平在分布式计算标准模型中几乎是不可能达到的。【32】一些最近的文章显示正确、有效并且对时间过程有意识的区块链可以促进资金公平，抵制中止方。【23,43,17】特别地，当合同倒退时，区块链可以让中止方失去一笔保证金，这笔保证金将归诚信方所有。

3.3 符号协议

我们现在解释一些编写合同的符号协议。网上完整版本【42】给出了一个前例。

• 货币及分类账：我们使用分类账【P】表示全球分类账中 P 方的账目。为了能更清晰，以$符号开头的变量表示货币，否则就表示普通变量。不像以太币中 Serpent 语言，在我们正式的记号中，当一个合同从 P 方收到一些金额$amount，这只是信息传输，此时还没有货币交易。货币交易只有在合同在分类账上操作时才会生效。

• 假名：各方可以使用假名获得更好的隐私权。特别是一方可以生成任意多个公钥。在我们的符号系统中，当我们提到 P 时，P 代表的是这一方的假名。我们采用的正式的区块链模型【43】提供了一种合同包装，能够管理假名的产生，并且信息签名对于建立一个合同认证通道来说是必需的。这些细节从主要合同程序中被抽离出来。

• 计时器：时间循环记录。在每一回合的开始，合同的计时器功能将被调用。变量 T 将当前时间编码。

• 接入点和变量范围：一个合同可以有很多接入口，每一个都会在接收到相应信息类型时被调用。因此，接入点的行为就像函数在接受到信息时被调用一样。所有的变量都是全球范围的，但是以下除外：当一个接入点反应“收到了从某一方 P 发来的信息”，就允许并登记了新一方 P。通常，合同对于任意一方来说都是公开的，任意一方都可以和它交互。当收到一条来自 P 的信息后（没有关键词“某一”），P 方就表示固定的一方-而一个正常的合同已经定义了 P。

这个符号系统【43】被设计出来不仅仅是为了方便，也是为了是系统更加精确，具有和通用可组合框架【28】兼容的形式意义。读者可以参照【43】来了解正式模型细节。虽然我们的网上完整版【42】证据依赖于这种形式主义，主干却可以独立理解。

4、以泄露机密信息为目的的 CSCs

作为证明智能合同力量的第一个例子，我们将

展示一个现有形式的对比特币部署的恶意合同是如何变得更加强大，在功能上加强为智能合同，并且在以太币系统中实现。

在比特币刺激的非法行为中有一种支付引发的泄密，即秘密的公开曝光。最近建立的网站 Darkleaks【3】作为一种为众筹公共泄密而存在的分散式市场，包括“好莱坞电影、商业机密、政府机密、专有源代码、医学或者国防类的工业设计等”。直观地说，我们在这一背景下定义佣金公平意味着如果承包商 C 在一个特定时间限制内完全泄露了指定机密，它将收到报酬。正如我们所知道的，Darkleaks 着重强调了比特币无法支持佣金公平。而我们认为 CSC 很可能实现佣金公平。

4.1 黑泄密

在黑泄密系统中，承包商 C 希望出售 M 组成的一部分，并将其分割为 n 部分 $\{m_i\}_{i=1}^{n}$。在一个时间点，T_{open} 被 C 预先设定，一个随机选取的 k 元子集 $\Omega \subset [n]$作为一个样例公开披露以诱惑捐赠者（购买者）——这些人将为公共泄露购买 M。当 C 确定所有捐赠者都已经支付了足够的费用，C 就会解密剩下的部分。三个参数（n,k,Topen）是由 C 设定的（n = 100，K = 20 是推荐的默认值）。

为了保证 M 在各方没有直接交互情况下的交易公平，黑泄密在比特币脚本语言中使用了一个（智能）协议。主要思想是对于 M 的一个给定部分 m_i，它并没有在 Ω 中作为样例出现，捐赠者向比特币账户 a_i 中用公钥 pki 支付一笔费用。m_i 被密钥 κ = H(pki)加密（H = SHA-256）。为了从账户 a_i 中支付这笔酬金，C 被比特币交易协议逼迫揭露 pki；因此自动支付酬金的行为让社会能够解密 m_i。

我们将在网上完整版本【42】中给出更多细节。

缺陷与弱点：黑泄密协议有三个主要缺陷，它们来自比特币脚本语言在无直接沟通情况下构建合同的基础功能限制。前两个破坏了佣金公平，第三个限制了功能。

1、**延迟释放：**C 可以阻止消费者或者捐赠者支付报酬或者释放未开封的 M 的部分，指导 M 失去价值。比如说，C，可以在电影上映后扣留其中的工业设计部分直到它被制作出来。

2、**选择性保留：**C 可以选择放弃被选部分的支付并且不公开它们。比如，C 可以泄露影片并且在最后的几分钟收集付款（这些款项很有可能不出现在样例 Ω 中），这将会显著减少泄露部分的价值。

3、**公开泄密：**黑泄密只能公开泄密。它不支持私人泄密的平等交易，也就是说，不支持购买者 P 为交换被公钥加密的机密 M 所支付的酬金。

另外，黑泄密有基础的协议上的漏洞：

4、**酬金偷窃：**在黑泄密协议中，pki 对应的比特币私钥 ski 来自于 m_i；ski = SHA-256(m_i)。因此，M 的资源（比如泄漏影片的持有者）可能得到 ski 并且盗取 C 收到的报酬。（当 C 索要报酬时，接收交易的恶意节点可以解密 m_i，并计算 ski = SHA-256(m_i)，在竞争性交易中用刷新网络的手段暗中盗取酬金【36】。）

最后一个问题可以通过用假名或随机生成部分加密密钥集合 $\{k_i\}_{i=1}^{n}$ 轻松解决，我们在 CSC 设计中就是这样做的。

注：在任意一个物品都可以用随机样例表示的协议中，C 可以向 M 中插入一组小数量的无价值或者重复的片段。这有一定几率不会导致生成非法样例 Ω，所以 Ω 只需要提供 M 全球有效性的微小证明。k 和 n 越大，受到这种攻击的可能就越小。这种类型的人类验证证明的正式分析和（或）使它们自动化的方法是不在本文讨论范围之内的一个有趣议题，但是在评定这种 CSC 的头尾连接安全性时非常重要。

4.2 一般性公共泄露 CSC

我们现在提出一种能用黑盒加密技术实现公共泄密的智能合同。（稍后会提出有效的实现方式。）这个合同能够用在预先指定的时间 T_{end} 强制泄露 M 或立即返还购买者的钱来克服黑泄密协议中的限制 1。它也用确保 M 以全有或全无的方式显露出来解决了限制 2（选择性扣留）。（我们将稍后解释如何实现私密泄露并且克服限制 3。）

我们再一次思考 C 在揭露样例部分 M^* 后以公开发行的方式售卖 M 的设定。

非正式协议描述：协议包括以下几个步骤：

•**创建合同：**卖方 C 通过加密随机生成的主密钥 msk

初始化一个智能合同。主密钥用来生成用于 $\{m_i\}_{i=1}^n$ 的（对称）加密密钥。C 向合同提供了 msk 的加密承诺 c0 := Enc(pk, msk, r0)。（为了满足我们安全证明的狭窄技术要求，这个承诺是一种在可信用安装过程中创造出来的公钥 pk 具有随机数 r0 时的加密过程。）主密钥 msk 可以用来解密 M 的所有泄露部分。

• **上传加密数据**：对于每一个 i ∈ [n]，C 都会生成一个加密密钥 k_i ：=PRF(msk, i)，并把第 i 个部分用 $ct_i = enc_{k_i}[m_i]$ 加密。C 向合同发送所有的加密部分 $\{ct_i\}_{i\in[n]}$（或者为了高效提供存储器中存储的哈希副本，比如点对点网络）。感兴趣的购买者/捐赠者可以下载 M 的部分，但是不能解密。

• **挑战**：合同在今天的以太币实践中产生了一个随机挑战集合Ω ⊂ [n]。另一种未来的可能性是某些有名的随机资源，比如 NIST 随机信标【9】，可能通过经过验证的数据反馈进行转播。

• **反馈**：C 向合同揭露了集合 $\{k_i\}_{i\in\Omega}$，并提供了 ZK 证明被揭露的密钥 $\{k_i\}_{i\in\Omega}$ 是从 msk 中正确生成并加密为 c_0 的。

• **收集捐赠**：捐赠期间，潜在的购买者/捐赠者可以使用已揭露的密钥 $\{k_i\}_{i\in\Omega}$ 解密相应的部分。如果他们喜欢解密片段，他们就可以向合同进行捐赠来为泄密行动做贡献。

• **接受**：当收集到了足够多的资金时，C 就会回收 msk（用 msk 发送随机密文）。如果合同成功地验证了承诺解除行为，所有捐赠的钱都会支付给 C。因此，合同强制执行了关于 msk 的公平交易来获取资金。（如果合同在 T_{end} 时间到期时还没有解密 msk，所有的捐赠都会被返还。）

我们提出的为实施公共泄密协议的 CSC PublicLeaks 在图 2 中给出。相应的用户侧已经在上文中解释过（并且可从合同中推知）。

Contract PublicLeaks

Init: Set state := INIT, and donations := {}. Let crs := $\mathsf{KeyGen}_{\mathrm{nizk}}(1^\lambda)$, pk := $\mathsf{KeyGen}_{\mathrm{enc}}(1^\lambda)$ denote hard-coded public parameters generated through a trusted setup.

Create: Upon receiving ("create", c_0, $\{\mathsf{ct}_i\}_{i=1}^n$, T_{end}) from some leaker $\mathcal{C}$:
Set state := CREATED.
Select a random subset $\Omega \subset [n]$ of size k, and send ("challenge", Ω) to $\mathcal{C}$.

Confirm: Upon receiving ("confirm", $\{(\kappa_i, \pi_i)\}_{i\in\Omega}$) from $\mathcal{C}$:
Assert state = CREATED.
Assert that $\forall i \in S$: π_i is a valid NIZK proof (under crs) for the following statement:
$$\exists(\mathsf{msk}, r_0),\ \text{s.t.}\ (c_0 = \mathsf{Enc}(\mathsf{pk}, \mathsf{msk}, r_0)) \wedge (\kappa_i = \mathsf{PRF}(\mathsf{msk}, i))$$
Set state := CONFIRMED.

Donate: Upon receiving ("donate", $amt) from some purchaser $\mathcal{P}$:
Assert state = CONFIRMED.
Assert ledger[$\mathcal{P}$] ≥ $amt.
Set ledger[$\mathcal{P}$] := ledger[$\mathcal{P}$] − $amt.
donations := donations ∪ {($amt, $\mathcal{P}$)}.

Accept: Upon receiving ("accept", msk, r_0) from $\mathcal{C}$:
Assert state = CONFIRMED
Assert c_0 = Enc(pk, msk, r_0)
ledger[$\mathcal{C}$] := ledger[$\mathcal{C}$] + sum(donations)
Send ("leak", msk) to all parties.
Set state := ABORTED.

Timer: If state = CONFIRMED and $T > T_{\mathrm{end}}$: $\forall(\$amt, \mathcal{P}) \in$ donations: let ledger[$\mathcal{P}$] := ledger[$\mathcal{P}$]+$amt. Set state := ABORTED.

图 2：合同 PublicLeaks-向公众泄露机密 M 来交换捐赠资金。

4.3 佣金公平：正式定义及证明

在网上完整版本中【42】，我们给出了作为理想功能的以公共泄密为目的的佣金公平的正式定义。我们也证明了 PublicLeaks 实现了这种功能，假定所有被解密的部分都是有效的——PublicLeaks 中的随机样例很有可能推动的一种特性。

4.4 优化及以太币实验

正式指定的合同 PublicLeaks 以黑匣方式使用了通用密码原语。我们现在给出一种现实的最优版本，这种版本依赖于随机预言模型（random oracle model,ROM），消除了可信的安装程序，也在以太币系统中实现了更高的效率和更简单的整合。【59】

现实最优版本：在创建合同的过程中，C 对 i∈[n] 选择了随机 $\kappa_i \xleftarrow{\$} \{0,1\}^\lambda$，并计算

$$c_0 := \{H(\kappa_1, 1), \ldots, H(\kappa_n, n)\}.$$

主密钥为 msk:=$\{k_1,\ldots,k_n\}$，即散列预图像集。

正如在 PublicLeaks 中一样，每个部分 m_i 都会以

$\mathrm{ct_i} = \mathrm{enc_{k_i}}[m_i]$的方式加密。（由于一些技术原因，这里$\mathrm{enc_{k_i}}[m_i] = \mathrm{m_i} \oplus$ [H(κ i, 1, “enc”)|| H(κ i, 2, “enc”)...,|| H(κ i, z, “enc”)]（对于足够大的 z）。

C 向智能合同提交$\mathrm{c_0}$。当被集合 Ω 挑战时，C 向合同透露$\{\mathrm{k_i}\}_{\mathrm{i}\in\Omega}$，并且随后用哈希证明了它的正确性，并将它与$\mathrm{c_0}$比较。为了得到捐赠，C 会透露整个 msk。

这种最优方案比我们一般的黑盒构建 PublicLeaks 要效率更低——因为主密钥直线先定了部分 n 的数量。但是对于典型的现实文件集大小（比如 n=100，像黑泄密所适合的），它更加高效 。

以太币基础上的实现：为了论证用现有技术实施泄密合同的可能性，我们用 Serpent 合同语言【10】在以太币系统内实现了一种 PublicLeaks 合同的版本【59】。我们在网上完整版本中详述了整个实现过程。【42】

我们实现的版本依赖于上述优化。作为一个技术问题，以太币现在并没有出现支持定时功能，所以我们实现了这样一种方式，购买者/捐赠者明确取款的时间，而不是接受自动退款，让定时器以这种方式工作。

这个公共泄密以太币合同效率很高，因为它不需要复杂的加密操作。它主要依赖哈希 (SHA3-256) 来生成随机数，并且验证哈希的承诺。存储条目以及哈希的操作的总数（用于加密密钥）是 O(n)，其中和黑泄密一样，推荐 n=100。（一个哈希函数实际调用需要几微秒，比如，在 Core i7 处理器上测量为 3.92 微秒。）

4.5 延伸：私人泄密

如上所述，黑泄密的缺陷 3 是无法支持私人泄密，其中 C 专门向购买者 P 售卖机密。在网上完整版本中【42】，我们展示了 PublicLeaks 是如何为了这个目的被修改的。基本思想是 C 不要直接揭露 msk，而是向购买者 P 提供密文$\mathrm{ct} = \mathrm{enc_{pk_P}}[msk]$，以及证明 ct 是正确生成的。我们描述一个黑盒变体，它的安全性可以用和 PublicLeaks 一样的方法证明。我们还描述了一种实际变体，它结合了 Chaum 和 Pedersen 的可验证随机函数（verifiable random function,VRF）【30】（$\{\mathrm{k_i}\}_{\mathrm{i=1}}^{\mathrm{n}}$的生成）以及 Camensich 和 Shoup 的可验证加密（verifiable encryption,VE）计划【27】（证明 ct 的正确性）。这个变体在现今能通过以太币的大数据运算支持被有效地利用。

Contract KeyTheft-Naive

Init: Set state := INIT. Let crs := $\mathsf{KeyGen}_{\mathrm{nizk}}(1^\lambda)$ denote a hard-coded NIZK common reference string generated during a trusted setup process.

Create: Upon receiving (“create”, \$reward, $\mathsf{pk}_\mathcal{V}$, T_{end}) from some contractor $\mathcal{C} := (\mathsf{pk}_\mathcal{C}, \ldots)$:
- Assert state = INIT.
- Assert ledger[$\mathcal{C}$] $\geq$ \$reward.
- ledger[$\mathcal{C}$] := ledger[$\mathcal{C}$] − \$reward.
- Set state := CREATED.

Claim: Upon receiving (“claim”, ct, π) from some purported perpetrator $\mathcal{P}$:
- Assert state = CREATED.
- Assert that π is a valid NIZK proof (under crs) for the following statement:
  $\exists r, \mathsf{sk}_\mathcal{V}$ s.t. $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}_\mathcal{C}, (\mathsf{sk}_\mathcal{V}, \mathcal{P}), r)$ and $\mathsf{match}(\mathsf{pk}_\mathcal{V}, \mathsf{sk}_\mathcal{V}) = \mathtt{true}$
- ledger[$\mathcal{P}$] := ledger[$\mathcal{P}$] + \$reward.
- Set state := CLAIMED.

Timer: If state = CREATED and current time $T > T_{\mathrm{end}}$:
- ledger[$\mathcal{C}$] := ledger[$\mathcal{C}$] + \$reward
- state := ABORTED

图 3：有缺陷的密钥盗窃合同（缺少佣金公平）

5、密钥和解 CSC

在本文介绍中例 1b 描述了一种 CSC，它奖励了犯罪者 P 将从受害人 V 处盗窃的密钥 skV 传递给 C 的行为——在这种情况下，受害者 V 是证书认证机构（CA）和公钥 pkV。C 生成了一个私钥/公钥加密对（skC，pkC）。合同接受这个结果，并由 P 宣布一个数对（ct，π）。如果 π 能有效证明$\mathrm{ct} = \mathrm{enc_{pk_c}}[sk_v]$，以及$sk_v$是和$\mathrm{pk}_v$对应的私钥，那么它向 P 支付报酬\$reward。

直观地说，如果一个密钥盗窃合同向犯罪者 P 支付传递密钥的报酬，且满足：（1）P 负责偷窃；（2）在相当长的一段时间内有效。那么这个合同达到佣金公平。

这种形式的合同可以被用于盗窃任何类型的私钥，比如 CA 的签名密钥、SSL/TLS 证书私钥，PGP 私钥等。（相似的合同可以请求滥用，但是不能完全解除私钥，比如仿制证书。）

图 3 显示了在我们的智能合同记号法中例 1b 中的合同。我们让这里的 crs 为 NIZK 计划捐赠一条常见的字符引用串，同时（pkV，skV）捐赠一种运算

法则，能够验证在目标公钥加密系统中 skV 是否是和某一公钥 pkV 相应的私钥。

如上所述，这个 CSC 不具备佣金公平。因此我们称它为 KeyTheft-Naive。我们将 KeyTheft-Naive 当作对激发和理解稍后将提出的佣金公平合同——KeyTheft 的构造很有帮助的切入点。

5.1 KeyTheft-Naive 的缺陷

合同 KeyTheft-Naive 无法实现佣金公平是由于以下两个不足：

撤销-认领攻击（Revoke-and-claim attack）：证书认证机构 V 可以撤销密钥 skV 并且自己提交支付密钥。CA 不仅仅是否定了合同的价值，而且从中获利。这种撤销-认领攻击显示 KeyTheft-Naive 从确保有用私钥 skV 的传递这个意义上来说不是佣金公平的。

突袭攻击（Rushing attack）：另一种攻击是突袭攻击。在第三节中我们提到，竞争对手可以任意重排信息——这是一种加密货币系统中对抗网络层可能攻击的反映。（也可参见正式区块链模型【43】。）因此，假如有一个从犯罪者 P 处得到有效索赔的情况，无信用的 C 就可以解密 skV，构造另一个看起来有效的声明，并让自己的索赔在原本的索赔之前到达。

5.2 在 KeyTheft-Naive 中修复漏洞

我们现在将展示如何调整 KeyTheft-Naive 来避免以上两种攻击并且实现佣金公平。

防御撤销-认领攻击：在对抗 KeyTheft-Naive 撤销-认领攻击中，V 抢先撤销公钥 pkV 并且用一个新的公钥 pk0V 替换它。如上所述，受害者可以扮演犯罪者 P，向合同提交 skV 并要求报酬。结果就是 C 向 V 支付了$reward 并获得了一把原来的密钥。

为了解决这个问题，我们向合同中加入了一种叫做奖励截断的特性，与合同接受撤销的证据 Π_{revoke} 相一致。

Π_{revoke} 可以是一种线上证书状态协议（Online Certificate Status Protocol，OCSP）的反馈，标志着 pkV 已经失效，有效的是在合同创建时还未知的（并且因此没有产出在合同中的）V 的新证书，或者一个包含 pkV 证书的证书撤销列表（CRL）。

C 可以提交 Π_{revoke}，但是为了让 C 参与的交互最小化，KeyTheft 可以向第三方提交者提供一种报酬$smallreward。这个报酬可以很少，因为 Π_{revoke} 对于原有用户来说很容易获得。

合同提供一种报酬，这种报酬建立在密钥 skV 仍然有效的间隔时间的基础上。令 T_{claim} 捐赠提供密钥 skV 的时间，并且 T_{end} 就是合同逾期时间（它不能超过包含目标密钥的证书的到期时间）。令 T_{revoke} 为 Π_{revoke} 被呈现的时间（如果在 T_{end} 之前没有发生撤销，那么 $T_{\text{revoke}} = \infty$ ）。合同分配给 P 属于 f(reward, t)的一份报酬，此时 t = min(T_{end} ， T_{revoke}) － T_{claim} 。

我们不在这里探究 f 的选择。我们注意到，尽管 CA 密钥 skV 可以用来仿制急用证书，比如恶意软件或者伪造软件更新，它的绝大部分价值可以在我们用 δ 捐赠的短时间内就被实现。（倾向正面实现价值一般来说是很常见的。【24】）一个合适的奖励函数应当是前载并且快速衰减的。这个特点的一个自然简单的选择是：

$$f(\$\text{reward}, t) = \begin{cases} 0 & : t < \delta \\ \$\text{reward}(1 - ae^{-b(t-\delta)}) & : t \geq \delta \end{cases}$$

这是对于 a<1/2 以及一些正实数 b 来说的。注意到报酬的大部分是在 t ≥ δ 时支付的。

防御突袭攻击：为了防御突袭攻击，我们将要求报酬分为两个阶段。在第一个阶段中，P 表达了提交真实报酬信息的承诺。然后 P 等待下一轮兑现承诺的时间并且揭露报酬信息。（由于证明中的技术细节问题，这个承诺必须适应安全；在证明中，模拟器必须能够在不知道被关注的字符串以及能够向任何字符串声明承诺的情况下模拟一个承诺。）在现实分散式加密货币系统中，P 可以在开启承诺之前等待多重区间间隔，来拥有更高的自信相信区块链不会分叉。在我们的这个形式中，一个圆可以对应一或多个区块间隔。

图 4 给出了一个密钥盗窃合同 KeyTheft，它能够防御撤销-认领攻击以及突袭攻击。

5.3、目标及状态暴露

KeyTheft-Naive 的一个不良特性是它的目标/受害者以及状态是公开可见的。V 因此可以了解它是否是 KeyTheft-Naive 的攻击对象。V 也可以观察成功的报酬声明——即 skV 是否被窃——因此可以采取

相应的防御行动。比如，由于密钥撤销需要大量资金以及时间，V 可以等待一个成功地稿酬声明出现，然后才执行撤销-认领攻击。

为了限制目标以及状态的暴露，我们注意到两种可能的对于 KeyTheft 的加强方式。第一种是多目标合同，在这个合同中，要求对多目标被害者中的任意一名实行密钥盗窃。第二种是我们所说的掩盖认领，用假的认领掩饰真正的认领行为。像在图 4 中所明确的，KeyTheft 的实施是一种多目标合同，这种技术同时提供了部分目标和部分状态的隐瞒手段。

多目标合同（Multi-target contract）：多目标合同征求任意 m 的潜在受害者 V1，V2，…，Vm。有许多设定，在这些设定中，不同受害者的私钥拥有相似的价值。比如，多目标合同 KeyTheft 可以为任何能解决 SSL/TLS 证书信任问题的 CA 的私钥 skV 提供报酬，被信任的有 Internet Explorer（其中超过 620【370】）等。

这里遇到的一个问题是合同状态是公开的，因此合同必须能够在不知道装备了哪个私钥——即不知道 i 的情况下验证有效认领的证明（私钥）skVi。我们的实验说明了，构建像 zk-SNARKs 这样的证明也是现实的。（承包商 C 本身可以通过解密 skVi 轻松掌握 i，并且生成 pkVi，确认相应的受害者。）

掩盖认领（Cover claims）：由于合同的状态是公开可见的，受害者 V 也能了解到是否有成功的认领已经提交给了 KeyTheft-Naive。这在单目标合同的情况下尤其不确定。

延迟 π 的提交（以及报酬支付）直到 T_{end} 是可能的，而并非用 ct 发送 NIZK 证明 π。（也就是说，认领需要输入（“claim”，ct）。）这种方法掩盖了 ct 的有效性。注意即使不需要 π，C 也能利用 ct。

能够支持这种掩盖的合同也能支持我们所说的掩盖认领。掩盖认领是无效要求的一种形式，即 ct 不是 skV 的一种有效加密。掩盖认领可能有 C 提交，掩饰了合同的真正状态。因此 C 不需要在创建合同后与之交互，合同可以在 T_{end} 时将报酬分成小部分并交与第三方提交掩盖认领。我们不在 KeyTheft 的版本中实施掩盖认领，也不在图 4 中讨论。

Contract KeyTheft

Init: Set state := INIT. Let crs := $\mathsf{KeyGen}_{\text{nizk}}(1^\lambda)$ denote a hard-coded NIZK common reference string generated during a trusted setup process.

Create: Same as in Contract KeyTheft-Naive (Figure 3), except that an additional parameter ΔT is additionally submitted by $\mathcal{C}$.

Intent: Upon receiving (“intent”, cm) from some purported perpetrator $\mathcal{P}$:
- Assert state = CREATED
- Assert that $\mathcal{P}$ has not sent “intent” earlier
- Store cm, $\mathcal{P}$

Claim: Upon receiving (“claim”, ct, π, r) from $\mathcal{P}$:
- Assert state = CREATED
- Assert $\mathcal{P}$ submitted (“intent”, cm) earlier such that cm = comm(ct||π, r).

Continue in the same manner as in contract KeyTheft-Naive, except that the ledger update ledger[$\mathcal{P}$] := ledger[$\mathcal{P}$] + $reward does not take place immediately.

Revoke: On receive (“revoke”, Π_{revoke}) from some $\mathcal{R}$:
- Assert Π_{revoke} is valid, and state $\neq$ ABORTED.
- ledger[$\mathcal{R}$] := ledger[$\mathcal{R}$] + $smallreward.
- If state = CLAIMED:
  - Let t := (time elapsed since successful **Claim**).
  - Let $\mathcal{P}$:= (successful claimer).
  - $\text{reward}_{\mathcal{P}} := f(\$\text{reward}, t)$.
  - ledger[$\mathcal{P}$] := ledger[$\mathcal{P}$] + $\text{reward}_{\mathcal{P}}$.
- Else, $\text{reward}_{\mathcal{P}} := 0$
- ledger[$\mathcal{C}$] := ledger[$\mathcal{C}$] + $reward −$smallreward − $\text{reward}_{\mathcal{P}}$
- Set state := ABORTED.

Timer: If state = CLAIMED and at least ΔT time elapsed since **Claim**:
- ledger[$\mathcal{P}$] := ledger[$\mathcal{P}$] + $reward;
- Set state := ABORTED.

Else if current time $T > T_{\text{end}}$ and state $\neq$ ABORTED:
- ledger[$\mathcal{C}$] := ledger[$\mathcal{C}$] + $reward.
- Set state := ABORTED.

// $\mathcal{P}$ should not submit claims after $T_{end} - \Delta T$.

图 4：密钥和解 CSC——能够防御撤销-认领攻击以及突袭攻击

5.4 佣金公平：正式定义和证明

我们对密钥盗窃定义佣金公平，它是一种在网上完整版本中【42】的理想功能，也为 KeyTheft 提供了一种正式的安全证明。

5.5 执行

我们依靠 zk-SNARKs 来高效实现上述协议。zk-SNARKs 是一种零知识证明，十分简易也很好验证。zk-SNARKs 的安全性比 UC-style 模拟证明所要求的更低。因此我们使用一种在 Hawk 的论文中【43】提到的通用变换，来提高安全性，使得零知识证明能够确保可提取的模拟公正。（简而言之，我们需要一次性密钥生成阶段来生成两把密钥：一把公共评价密钥，以及一把公共验证密钥。为了证明特定 NP 的陈述，不可信的证人使用评价密钥计算一个简易证明；任何验证着都可以使用公共验证密钥验证该证明。验证者在我们这个例子中就是合同。）在我们执行的过程中，假定生成密钥是具有信用的一方秘

密操作的。为了在密钥生成阶段使信用的重要性最小化，安全的多方计算技术可以像在【21】中一样被使用。

1-Target	*#threads*	RSA-2048	ECDSA_P256
Key Gen.[$\mathcal{C}$]	1	124.88 sec	242.30 sec
	4	33.53 sec	73.38 sec
Eval. Key		215.93 MB	448.24 MB
Ver. Key		6.09 KB	5.15 KB
Prove[$\mathcal{P}$]	1	41.02 sec	83.63 sec
	4	**15.7 sec**	**32.19 sec**
Proof		711 B	711 B
Verification	Contract	**0.0089 sec**	**0.0087 sec**

500-Target	*#threads*	RSA-2048	ECDSA_P256
Key Gen.[$\mathcal{C}$]	1	161.56 sec	263.07 sec
	4	43.35 sec	78.31 sec
Eval. Key		279.41 MB	490.85 MB
Ver. Key		4.99 KB	4.99 KB
Prove[$\mathcal{P}$]	1	54.15 sec	84.69 sec
	4	**23.54 sec**	**33.49 sec**
Proof		711 B	711 B
Verification	Contract	**0.0087 sec**	**0.0087 sec**

表 1：单目标和 500 目标合同中密钥和解 zk-SNARK 循环的实现。和计算工作的整体实现相关。

Zk-SNARK 循环：为了估计计算和验证所消耗的证明量，我们实施了上述盗窃协议 RSA-2048 和 ECDSA P256 密钥，它们在 SSL/TLS 证书中被广泛使用。这个循环有两个主要子循环：一个密钥检测循环和一个加密循环。加密循环是使用 RSAES-OAEP【41】和 2048 位密钥实现的。依托这些高级执行编译程序可以生成复杂的循环供 zk-SNARK 证明计算使用。我们建立了定制的循环发生器来生成更多高效循环。然后我们使用最先进的 zk-SNARK 书库【22】来获取评价结果。表 1 显示了循环的评价结果，分别针对单目标和多目标的合同。这个实验是在 Amazon EC2 r302xlarge 和 61GB 的内存空间、2.5GHz 处理器的条件下进行的。

结果产生了两个有趣的现象：1、当犯罪者获得 TLS 公钥时，计算 zk-SNARK 证明只需要不到两分钟，耗费不到 1USD【4】，对于单目标和多目标合同都是这样；2、使用 500 密钥的多目标合同在证人端最差情况下也只需要花费 13 秒。同时，合同的验证花费和单目标情况下基本上是一样的。这种情况的实现是因为高效 Merkle tree 循环，它证明了和解公钥在目标密钥串中的组成，并且使用了相同的单目标循环组成。

撤销证书的确认：上述合同的报酬功能依赖于证书撤销的时间，因此合同需要能够处理证书撤销证明的模块，比如 CRLs 和 OCSP 反馈，以及在它们上面验证 CA 数字签名。举个例子，我们测量 openssl verify-crl_check 指令的运行时间，测量撤销证书【12】和 2016.2.15，CRL 最后升级【8】，大小为 143KB。平均下来，验证过程在 2.3GHz i7 处理器上大概需要 0.016 秒。签名算法则是用 2048 位密钥将 SHA-256 和 RSA 加密。由于 OCSP 反馈会比 CRLs 更小，OCSP 的验证时间也可能会更短。

多目标合同的案例：验证但目标合同的撤销证明很简单：合同可以确定撤销证明是否和目标密钥相对应。而在多目标合同中，合同不知道哪一把目标密钥和提交的密钥盗窃证明 P 相对应。因此，我们需要一个证明来讲撤销和被窃密钥联系起来，并且它是由 C 提交的。

我们建立一个 zk-SNARK 循环，通过 C 可以证明犯罪者提交的密文和目标和解密钥之间的联系。为了更加高效，我们通过强迫 P 排除了对密钥检测子循环的需要来添加和解公钥在申请加密之前的秘密参数。表 2 中的评价说明了收到证明的合同所做的验证的效率，以及 C 构建证明的实际性。和认领的案例情况相反，这个循环的一次性密钥生成比如由 C 独立完成，因此 C 不能欺骗合同。我们注意到建立的撤销循环对于目标密钥加密系统来说是不变的。

	#threads	RSA-2048	ECDSA_P256
Key Gen.	1	124.64 sec	124.35 sec
	4	33.52 sec	33.38 sec
Eval. Key		215.41 MB	214.81 MB
Ver. Key		5.51 KB	4.88 KB
Prove[$\mathcal{C}$]	1	41.08 sec	40.96 sec
	4	**15.94 sec**	**15.59 sec**
Proof		711 B	711 B
Verification	Contract	**0.0087 sec**	**0.0086 sec**

表 2：和解密钥 zk-SNARK 循环的实现，目的为多目标合同案例中需要的撤销过程。和整个计算工作相联系。

6、Calling Card 犯罪

如上所述，分散式智能合同系统（比如以太币）拥有配套服务，比如提供经过验证的数据反馈、数字签名的认真消息、关于现实世界的事实等。尽管仍然处于起步阶段，它强更大的能力对于许多智能合同的应用来说是基础性的，也将很大程度上扩大 CSCs 的范围，其中包含现实世界的一些事件，如下所述：

EXAMPLE 2：（暗杀 CSC）：承包商 C 提出了一个合同 Assassinate 来暗杀参议员 X。合同将支付给犯罪者 P 一定报酬。

合同 Assassinate 接受犯罪者 P 输入的承诺 vcc，提前指定暗杀的细节（日期，时间，地点等）。为了获取报酬，P 在暗杀后回收 vcc。为了验证 P 的认领，Assassinate 搜索当前事件中的已验证数据反馈来证实参议员 X 的暗杀和 vcc 细节相匹配。

这个例子也说明了 Calling Card 犯罪的实施方法。“Calling Card”指的是待执行犯罪中不可预知的因素（比如，EXAMPLE 2 中的日期、时间、地点）。Calling Card 和已经过检验的数据反馈可以支持很多 CSCs 的通用框架。

CSC 的通用结构以下述 calling card 为基础。P 提前向合同 calling card cc 提供了承诺 vcc。在实施犯罪后，P 证明 cc 和 vcc 相互对应（比如回收 vcc）。合同指向一些可信的、已经经过检验的数据反馈证明：（1）犯罪已经实施（2）calling card cc 和犯罪相对应。如果两个条件都满足，合同就会向 P 支付酬金。

直观来说，我们定义佣金公平的意思是如果 P 对实施犯罪负有责任，那么他获得报酬。（正式的定义见网上完整版本【42】。）

更详细地说，令 CC 为一系列可能的 calling card 的集合，cc∈CC 捐赠了一个 calling card。如上所述，可以预见一个已经过检验的数据反馈生态系统将会出现在智能合同系统中。

我们将数据反馈视为一系列从 S 中获得的数对，其中 (s(t), σ(t))是时间 t 的输出。这里的价值 s(t) ∈ {0, 1}*是时间 t 释放数据的一部分，σ(t)是相应的数字签名；S 有一对相关的私钥/公钥（skS,pkS）用来完成/验证数字签名。

注意到 calling-card 合同一旦被创造，就不再从 C 要求更多交互，使得执法部门很难用随后的网络交通追踪 C。

6.1 实例：网站篡改合同

举个例子，我们指出一个简单的 CSC SiteDeface 被网站毁损。承包商 C 指定网站网址被黑客攻击，声明语句将显示。（比如，声明语句=“Anonymous. We are Legion. We do not Forgive...” 而网址 = whitehouse.gov.）

我们假设一种数据反馈认证网站的内容，即 s(t) = (w, url, t)，其中 w 是网页内容的表示，t 是一个时间戳，在合同时间内简便地表示时间。（为了提高效率，w 可能是指向网页内容的指针或哈希。）这样一种反馈可能会表现为比如说黑客网站存档的数字签名版本。功能 SigVer 指出了签名验证操作。

正如样例参数化，我们可以令 $CC = \{0,1\}^{256}$ 即 cc 是一个 256 位的字符串。犯罪者 P 选择一个 calling card $cc \xleftarrow{\$} \{0,1\}^{256}$ 以及承诺 vcc :=commit(cc, P; ρ)，其中 commit 表示一个承诺计划，并且 $\rho \in \{0,1\}^{256}$ 是一个随机字符串。（实际上，HMAC-SHA256 在以太币系统中是实施起来相对较简单的合适选择。）P 用揭露 commit 所有论点的方法进行回收。

CSC SiteDeface 见图 5。我们所用的例子是为理解清晰而简化的。我们假定在这个例子中，网页只包含了 calling card 和陈述，但是可能支持任意复杂的内容。

Contract SiteDeface

Init: On receiving ($reward, $pk_{\mathcal{S}}$, url, stmt) from some $\mathcal{C}$:
Store ($reward, $pk_{\mathcal{S}}$, url, stmt)
Set $i := 0$, $T_{\text{start}} := T$

Commit: Upon receiving commitment vcc from some $\mathcal{P}$:
Store $vcc_i :=$ vcc and $P_i := \mathcal{P}$; $i := i + 1$.

Claim: Upon receiving as input a tuple (cc, ρ, σ, w, t) from some $\mathcal{P}$:
Find smallest i such that $vcc_i = \text{commit}(cc, \mathcal{P}; \rho)$, abort if not found.
Assert $w =$ cc || stmt
Assert $t \geq T_{\text{start}}$
Assert $\text{SigVer}(pk_{\mathcal{S}}, (w, url, t), \sigma) = \text{true}$
Send $reward to P_i and abort.

图 5：网站篡改 CSC

注：SiteDeface 可以通过使 P 生成数字签名 cc 而选择性执行。不过我们的实验也容纳了很短的、低级加密 calling card cc，对于通用 calling-card CSCs 十分重要。参见网上完整版本【42】。

执行：给定一个已经过认证的数据反馈，执行 SiteDeface 就会变得直接而有效。主要的开销在于认领模块，其中合同计算了几个哈希并且在取回的网站数据中验证反馈的签名。像在第四节中提到的，调用哈希函数可以在很短的时间（4 微秒）内计算出结果，而确认签名需要花费更多的时间。比如，如果取回的内容有 100KB，合同只需要 10 毫秒来验证 RSA-2048 签名。

6.2 佣金公平：正式定义

我们给出一种 calling-card CSC 的佣金公平正式定义。【42】

我们不提供安全证明，因为这要求有现实世界系统模型，而这超出了本文的范围。

6.3 其他 calling-card 犯罪/数据反馈腐败

使用像 SiteDeface 之类的 CSC 时，承包商 C 可以征求其他犯罪，比如暗杀、袭击、蓄意破坏、劫机、绑架、拒绝服务攻击以及恐怖袭击。犯罪者 P 必须能够指明由已验证数据反馈得出的可靠的 calling card。

然而，一个自然对策对于一个以 calling-card 为基础的 CSC 来说是一个对手，即尝试使 CSC 无效的一方，破坏已验证的数据反馈或者数据资源防止他们提供无效数据。在涉及高位 CSCs 的案例中，这种方法是可行的。比如，如果有一个资金充足的 CSC 要求暗杀一名公职人员，数据反馈提供者或者信息资源可能会发布虚假的目标死亡报告。

相反，如果 C 关心在资源中心的信息抑制，他当然可以创造一个引用多个资源的 CSC，比如多个消息反馈。只有在某一特定部分报告事件时，CSC 才会将事件当作可信的。这种资源多样化将使得腐败变得更加困难（也是一个为良性合同提供防御腐败韧性的好办法）。

我们在网上完整版本中【42】讨论了一些有关于 calling-card CSCs 构建得到一般问题。

7、对策

我们进行工作的主要目的是强调在类似以太币的智能合同系统发展之际研制对抗 CSCs 对策的重要性。我们在这里简单讨论一下这个问题。

类似于黑名单“污点”硬币/交易的思想——那些已知罪犯使用过的——已经为加密货币提出了。在第二节提到的积极替代是一种身份托管思想，在更早的（集中式）电子现金系统中有时被称为“受托人追踪”（trusteebased tracing）【26,58】。受托人追踪计划允许信用机构（“受托人”）或者这些机构的法定人追踪金钱交易，否则这些交易是保持匿名的。然而在分散式加密货币系统中，用户不向当局登记身份——许多用户反对这样做。但实际上，用户自愿注册并只选择接受他们认为可以注册的货币，这是有可能的。腐坏硬币的思想不太能被加密货币社会所接受，因为它破坏了基本的现金代替性能【14,48】，而受托人追踪也有相似的缺陷。目前还不清楚什么尸体应当被授予执行黑名单或登记用户的权力。

我们观察到，对于加密货币系统的大多数用和来说，监控和（或）抑制犯罪活动是经济有益的，因为犯罪活动可以降低接受度，因而影响市场价值。这种观察结果促进了发明新的方式。比如，Blockchain Alliance【15】的任务是打击区块链犯罪活动。相似地，加密货币系统的核心开发者已经表达了过滤区块链内容的期望，通过和仇恨言论审查类比的方式【54】。

将比特币交易视为犯罪是有挑战性的，因为交易本身不屑道任何有关支付环境的信息。相反，如果 CSCs 被视为犯罪（比如暗杀合同），那么它就是自诉对象。逆转 CSC 的二进制文件是很有挑战性的，但是我们注意到对于 CSCs 来说，为了提高效率——就像我们上述例子中一样——他们的部署者必须进行宣传，吸引别人注意到他们合同的特性。（比如，承包者需要进行暗杀，正在寻找暗杀者。）因此，我们的假设是一个足够警觉的加密货币社会能够发现很多 CSCs 的存在并且被激励过滤或者清除相关交易。一个简单的潜在机制是当矿工们发现被可信机构标识为 CSCs 的交易时，忽略它们。

还有一种更加积极的做法也是可能的，我们称之为中立受托人智能合同。智能合同系统可能是这样设计的：当局、法人，或者合适的通用系统参与者集合，被授权将合同从区块链中移除。这样一种方法相比传统的受托人保护策略有一个巨大的优势，它不需要用户登记身份。这个想法是否对加密货币社会的胃口，一个被广泛接受的当局能否被确定身份，当然，这是开放式的问题，就像正确的技术支持机制一样，没有答案。

总而言之，我们在这里所做的工作对于加密货币社会暴露在 CSCs 的威胁下这个事实是十分有价值的，使监测和合适的对策结构能够有序地被制定。我们相信创造一种在分散式智能合同生态系统发展早期的警觉意识是非常重要的。

8、结论

我们已经说明了一系列佣金公平的恶意智能合同（CSCs）在分散式智能合同货币系统中是现实的。我们提出了三种 CSC——机密泄露、密钥盗窃、calling-card 犯罪——并且说明了他们在智能合同系统以及现有加密技术的支持下能高效执行。合同 PublicLeaks 和它的私人变体现今可以在 Serpent——一种以太币脚本语言中高效执行。KeyTheft 只需要少量的已经设计好的支持 zk-SNARKs 的操作码就可以实现高效部署。Calling-card CSCs 可能会获取充足的数据反馈生态系统。肯定还有更多 CSCs 是可能实现的。我们强调智能合同在分布式加密货币系统中有许多有前景的、合法的应用，完全禁用智能合同既不合理也不可能。我们的工作所提出的最紧要的

问题是如何在支持智能合同有益应用的同时，建立防御系统来对抗智能合同的危险滥用。

9、参考文献

[1] http://www.smartcontract.com.

[2] http://koinify.com.

[3] https://github.com/darkwallet/darkleaks.

[4] Amazon EC2 pricing.
http://aws.amazon.com/ec2/pricing/.

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

[6] Bitcoin ransomware now spreading via spam campaigns.
http://www.coindesk.com/
bitcoin-ransomware-now-spreading-via-spam-campaigns/.

[7] bitoinj. https://bitcoinj.github.io/.

[8] CRL issued bby Symantec Class 3 EV SSL CA - G3.
http://ss.symcb.com/sr.crl.

[9] NIST randomness beacon. https://beacon.nist.gov/home.

[10] Serpent. https://github.com/ethereum/wiki/wiki/Serpent.

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

[12] Verisign revoked certificate test page.
https://test-sspev.verisign.com:
2443/test-SPPEV-revoked-verisign.html. Accessed: 2015-05-15.

[13] Zcash. Referenced Aug. 2016 at z.cash.

[14] Mt. Gox thinks it’s the Fed. freezes acc based on \tainted"
coins. (unlocked now).
https://bitcointalk.org/index.php?topic=73385.0, 2012.

[15] Blockchain Alliance. www.blockchainalliance.org, 2016.

[16] Ethereum and evil. Forum post at Reddit; url:
http://tinyurl.com/k8awj2j, Accessed May 2015.

[17] M. Andrychowicz, S. Dziembowski, D. Malinowski, and
L. Mazurek. Secure Multiparty Computations on Bitcoin.
In S & P, 2013.

[18] J. Bates. Trojan horse: AIDS information introductory
diskette version 2.0,. In E. Wilding and F. Skulason, editors, Virus Bulletin, pages 3{6. 1990.

[19] J. Bell. Assassination politics.
http://www.outpost-of-freedom.com/jimbellap.htm, 1995-6.

[20] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers,
E. Tromer, and M. Virza. Zerocash: Decentralized
anonymous payments from Bitcoin. In S & P. IEEE, 2014.

[21] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and
M. Virza. Secure sampling of public parameters for succinct
zero knowledge proofs. In S & P, 2015.

[22] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza.
Succinct non-interactive zero knowledge for a von Neumann
architecture. In USENIX Security, 2014.

[23] I. Bentov and R. Kumaresan. How to Use Bitcoin to Design
Fair Protocols. In CRYPTO, 2014.

[24] L. Bilge and T. Dumitras. Before we knew it: an empirical
study of zero-day attacks in the real world. In CCS, 2012.

[25] V. Blue. Cryptolocker’s crimewave: A trail of millions in
laundered Bitcoin. ZDNet, 22 December 2013.

[26] E. F. Brickell, P. Gemmell, and D. W. Kravitz.
Trustee-based tracing extensions to anonymous cash and
the making of anonymous change. In SODA, volume 95,
pages 457{466, 1995.

[27] J. Camenisch and V. Shoup. Practical verifiable encryption
and decryption of discrete logarithms. In CRYPTO ’03. 2003.

[28] R. Canetti. Universally composable security: A new
paradigm for cryptographic protocols. In FOCS, 2001.

[29] D. Chaum. Blind signatures for untraceable payments. InCRYPTO, pages 199{203, 1983.

[30] D. Chaum and T. P. Pedersen. Wallet databases

with
observers. In CRYPTO’92, pages 89{105, 1993.
[31] N. Christin. Traveling the Silk Road: A measurement
analysis of a large anonymous online marketplace. InWWW, 2013.
[32] R. Cleve. Limits on the security of coin flips when half the
processors are faulty. In STOC, 1986.
[33] G. Danezis, C. Fournet, M. Kohlweiss, and B. Parno.
Pinocchio Coin: building Zerocoin from a succinct
pairing-based proof system. In PETShop, 2013.
[34] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi.
Step by step towards creating a safe smart contract:
Lessons and insights from a cryptocurrency lab.
https://eprint.iacr.org/2015/460.
[35] P. T. et al. Darkwallet on twitter: \DARK LEAKS coming
soon. http://t.co/k4ubs16scr". Reddit:
http://bit.ly/1A9UShY.
[36] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin
mining is vulnerable. In FC, 2014.
[37] E. F. Foundation. EFF SSL observatory. URL:
https://www.eff.org/observatory, August 2010.
[38] A. Greenberg. ’Dark Wallet’ is about to make Bitcoin
money laundering easier than ever.
http://www.wired.com/2014/04/dark-wallet/.
[39] A. Greenberg. Alleged silk road boss Ross Ulbricht now
accused of six murders-for-hire, denied bail. Forbes, 21
November 2013.
[40] Intel. Intel software guard extensions programming
reference. Whitepaper ref. 329298-002US, October 2014.
[41] J. Jonsson and B. Kaliski. Public-Key Cryptography
Standards (PKCS) #1: RSA Cryptography Specifications
Version 2.1, 2003. RFC 3447.
[42] A. Juels, A. Kosba, and E. Shi. The ring of gyges:
Investigating the future of criminal smart contracts.
Cryptology ePrint Archive, Report 2016/358, 2016.
http://eprint.iacr.org/2016/358.
[43] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou.
Hawk: The blockchain model of cryptography and
privacy-preserving smart contracts. In S & P. IEEE, 2016.
[44] V. Kotov and M. Rajpal. Understanding
crypto-ransomware. Bromium whitepaper, 2014.
[45] A. Krellenstein, R. Dermody, and O. Slama. Counterparty
announcement.
https://bitcointalk.org/index.php?topic=395761.0, January
2014.
[46] R. Kumaresan and I. Bentov. How to Use Bitcoin to Incentivize Correct Computations. In CCS, 2014.
[47] D. Mark, V. Zamfir, and E. G. Sirer. A call for a temporary
moratorium on \The DAO" (v0.3.2). Referenced Aug. 2016
at http://bit.ly/2aWDhyY, 30 May 2016.
[48] J. Matonis. Why Bitcoin fungibility is essential. CoinDesk,
1 Dec. 2013.
[49] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko,
D. McCoy, G. M. Voelker, and S. Savage. A fistful of
bitcoins: characterizing payments among men with no names. In IMC, 2013.
[50] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin:
Anonymous Distributed E-Cash from Bitcoin. In S & P, 2013.
[51] M. Moser, R. Bohme, and D. Breuker. An inquiry into
money laundering tools in the bitcoin ecosystem. In eCRS,
2013.
[52] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash
System. http://bitcoin.org/bitcoin.pdf, 2009.
[53] R. Pass and a. shelat. Micropayments for

peer-to-peer
currencies. Manuscript.
[54] M. Peck. Ethereum developer explores the dark side of
Bitcoin-inspired technology. IEEE Spectrum, 19 May 2016.
[55] K. Poulsen. Cybercrime supersite ‘DarkMarket’ was FBI
sting, documents confirm. Wired, 13 Oct. 2008.
[56] D. Ron and A. Shamir. How did Dread Pirate Roberts
acquire and protect his bitcoin wealth? In FC. 2014.
[57] S. V. Solms and D. Naccache. On blind signatures and
perfect crimes. Computers Security, 11(6):581{583, 1992.
[58] M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind
signatures. In Eurocrypt, pages 209{219, 1995.
[59] G. Wood. Ethereum: A secure decentralized transaction
ledger. http://gavwood.com/paper.pdf, 2014.
[60] A. Young and M. Yung. Cryptovirology: Extortion-based
security threats and countermeasures. In S & P, 1996.
[61] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi.
Town Crier: An authenticated data feed for smart
contracts. In ACM CCS, 2016. (To appear.).

3170101522
叶一苇
周二 9、10