10 篇塑造现代零知识证明的必读论文

来源：登链社区

零知识证明在近 40 年中取得了显著的发展，达到了前所未有的复杂性和效率水平。如今，每天都有新的论文和项目涌现，建立在丰富的思想和创新基础之上。

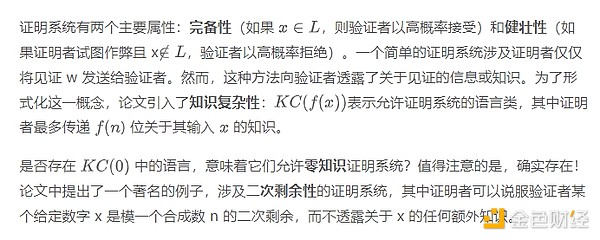
想知道这一切是如何开始的吗？ 在这篇文章中，我们将深入探讨零知识证明的历史，探索 10 篇帮助塑造这一领域的里程碑论文。

## 1 - 起源

Goldwasser, Micali, Rackoff - 交互式证明系统的知识复杂性 (1985) [^1]

我们的第一个里程碑带我们回到 1985 年那篇开创性的论文！这项工作引入了许多术语和基础概念，这些概念至今仍然是零知识证明的核心。

首先，论文定义了一个证明系统，其模型为一个涉及两个概率图灵机的双方协议：一个证明者和一个验证者。证明系统的目标是使证明者能够说服验证者某个给定输入 x 属于正式语言 L。在大多数早期工作中，证明者是计算上不受限制的，而验证者则限制在多项式时间计算。在交互结束时，验证者输出“接受”或“拒绝”。



## 2 - 第一个应用

Fiat, Shamir - 如何证明自己：识别和签名问题的实用解决方案 (1986) [^2]

这篇由 Fiat 和 Shamir 撰写的论文，发表于零知识证明基础工作的一年后，介绍了这些概念的第一个实际应用。他们提出了两个协议：一个识别方案，这是交互式的，另一个是签名方案，这是非交互式的。两者之间的关键区别在于，在识别方案中，第三方可以通过制作有效的记录来说服自己一个虚假的陈述。而在签名协议中，即使是一个不诚实的证明者也无法说服自己一个虚假的陈述，从而使签名不可伪造。

识别方案将二次剩余性证明系统应用为交互式协议，其中验证者发送随机挑战，证明者相应地回应。签名方案通过用对哈希函数的调用替换验证者的挑战来修改这一点。

作者的名字听起来熟悉吗？这是一个强大通用技术的首次实例，现在被广泛称为Fiat-Shamir 启发式。它使得通过用对随机预言机的查询（在实践中是加密哈希函数）替换验证者的挑战，将任何公共币交互式证明系统转换为非交互式的。

## 3 - 我们究竟能证明什么？

Goldreich, Micali, Wigderson - 如何在零知识中证明所有 NP 语句及密码协议设计的方法论 (1987) [^3]

这篇 1986 年的论文给出了一个显著的结果：每个 NP 中的语言都承认一个零知识证明系统。这很重要，因为这意味着我们可以在不透露额外信息的情况下证明任何可以在多项式时间内验证的陈述的真实性。作者通过提供一个图的 3-着色性的证明系统来展示这一点，该问题是确定图的节点是否可以用三种颜色着色，使得没有两个相邻的节点共享相同的颜色。此外，证明仅假设存在概率加密。

证明的直觉如下：在每一轮中，

通过以多项式次数运行此协议，验证者以高概率确信证明者知道有效的 3-着色，而不学习任何信息，因为打开的颜色在每一步都是随机的！

在这项工作中还有两篇其他论文同样值得注意：所有可证明的内容都可以在零知识中证明[^4]，该论文表明每个 IP 中的语言都有一个零知识证明系统，以及IP = PSPACE [^5]，表明 IP 与 PSPACE 一样强大。

## 4 - PCPs 和简洁非交互式证明的起源

Micali - 计算上可靠的证明 (2000) [^6]

这篇 2000 年的论文由 Micali 撰写，是零知识证明历史上的一个重要贡献。它甚至可以被视为第一个 SNARK 构造，尽管当时尚未创造出 SNARK 这个术语！

Micali 的构造将任何概率可检查证明（PCP）转化为简洁且非交互式的证明。PCP 是可以通过仅阅读少量位来验证的证明，而一个关键结果，即PCP 定理 [^7]，表明每个 NP 中的语言都有一个可以通过仅阅读证明中的常数位来验证的 PCP！

Micali 的构造使用默克尔树如下：

通过简单应用 Fiat-Shamir 启发式，可以使该构造变为非交互式（这种转换的一个交互版本是由 Kilian 提出的 [^8]）。论文还关注计算效率：实际上，验证者不需要接收整个证明，而只需接收常数数量的位和认证路径，这使得证明简洁。该系统的主要缺点是 PCP 构造不切实际，这促使了交互式预言机证明（IOPs）的发展，IOPs 是 PCP 的一种推广，可以产生实用的论证。

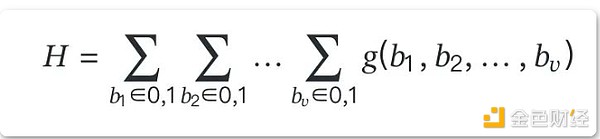
## 5 - 双重高效的 IPs

Goldwasser, Kalai, Rothblum - 委托计算：针对麻瓜的交互式证明 (2015) [^9]

这篇论文强烈关注效率，并引入了广为人知的GKR 协议，这是一个针对可用算术电路的公共币交互式协议。值得注意的是，验证者和证明者都在多项式时间内运行，使其成为双重高效的交互式证明。

该协议开始于证明者和验证者就一个输入扇入为 2 的算术电路达成一致。然后，证明者发送给定输入值的电路声称输出。协议通过逐层检查电路进行，从输出层开始，向输入层移动。每一步将当前层的声明减少为对前一层的声明，直到验证者到达输入层，在那里他们可以检查它是否与原始输入匹配。

在每一步中，主要使用的原语是求和检查协议 [^10]，这是一种交互式证明，使证明者能够说服验证者关于一个具有 oracle 访问权限的 v-变量多项式 g 的值之和，定义在布尔超立方体上：



由于其高效性和简单性，求和检查协议和 GKR 协议在实践中被广泛使用。有关进一步的解释，可以在 Thaler 的注释中找到这些协议的替代概述 [^11]。

## 6 - 第一个实用的 SNARK

Gennaro, Gentry, Parno, Raykova - Quadratic Span Programs and Succinct NIZKs without PCPs (2013) [^12]

我们现在跳到一篇介绍第一个实用 SNARK 构造的论文！这项工作标志着旨在创建不依赖于低效 PCP 定理的 SNARK 研究的巅峰。虽然 PCP 定理提供了一个理论上的 SNARK 构造，但对于实际应用来说速度太慢，因此研究人员试图寻找更高效的替代方案。例如，Groth 在 2010 年提出了一种基于双线性群和配对的非交互式论证系统 [^13] ，尽管它需要证明者的二次时间。然而，这篇论文实现了线性证明者时间，代表了实际应用的重大改进。

这项工作为其他重要协议铺平了道路，例如Pinocchio 协议[^14] ，以及最终著名的Groth16[^15] 证明系统。该论文还介绍了二次跨度程序和二次算术程序，这些构造在这些系统中仍然至关重要。

这些构造的一个显著缺点是需要可信设置，这意味着公共参考字符串生成阶段产生的秘密信息（通常称为有毒废物）如果不正确销毁，可能会被用来创建虚假证明。此外，这种设置不是通用的，这意味着每个电路都需要新的设置。尽管存在这些限制，生成的证明大小在不同构造中仍然是最小的，使其成为各种应用的热门选择。

还值得一提的是，Zerocash [^16] 的第一次迭代是一个早期且有影响力的区块链应用，利用了 zk-SNARKs，建立在这些系统之上。

## 7 - PlonK SNARK

Gabizon, Williamson, Ciobotaru - PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge (2019) [^17]

这篇 2019 年的影响力论文介绍了 PlonK SNARK，这是一个基于多项式交互式 oracle 证明（IOPs）的系统，这意味着验证者可以对一些多项式进行 oracle 访问，并可以在选择的点上对其进行评估。该系统使用各种多项式小工具来证明关于多项式的陈述，其中最显著的是大乘积论证，允许证明者表明在一个域上的评估的乘积为 1。利用这一点，我们可以构造一个置换论证来证明两个序列是彼此的置换。使用这些小工具，证明者可以为任何算术电路构造证明，验证者可以以非交互的方式验证它。

在实践中，oracle 访问是通过多项式承诺方案（PCS）实现的，这允许证明者：

这使得验证者可以在任何点查询多项式并验证 IOP 的关系。论文中建议的 PCS 是KZG 承诺方案 [^18] ，它既高效又实用。KZG 使证明者对多项式的承诺作为单个群元素，验证者可以通过计算几个椭圆曲线配对来确认开放值。虽然 KZG 需要可信设置，但它是通用的，可以在设置后用于任何电路。然而，PlonK 可以与其他 PCS 方案结合，使其适应透明论证系统。

此外，PlonK 中的置换论证启发了查找论证。查找论证使证明者能够表明一个序列的所有元素都包含在另一个序列中，这对于 zkVM 架构非常有用。查找论证允许将见证分解为更小的轨迹，并证明它们之间的查找关系，从而使复杂证明更高效。

## 8 - STARKs

Ben-Sasson, Bentov, Horesh, Riabzev - Scalable, transparent, and post-quantum secure computational integrity (2018) [^19]

这篇论文介绍了 STARK 证明系统，另一种流行的证明系统，基于FRI [^20] ，这是一个用于 Reed-Solomon 码的接近性测试的 IOP 协议。在 STARKs 中，证明者通过在一个域上构建默克尔树来承诺多项式的评估。由于承诺的值最初是未知的，验证者使用 FRI 确认这些评估形成一个足够低度的有效多项式。该协议还可以作为多项式承诺方案，允许验证者检查承诺多项式在任何点的评估。

STARKs 最引人注目的特点之一是它们仅依赖于密码学抗碰撞哈希函数，而不是其他密码学假设，如离散对数问题。这使得 STARKs 在潜在的后量子安全性方面具有优势，因为抗碰撞哈希函数被广泛认为即使在量子攻击下也安全。此外，STARKs 是透明的，即它们不需要任何可信设置。它们也是通用的，这意味着它们不局限于特定电路，在各种应用中提供灵活性。

## 9 - 递归

Valiant - Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. (2008) [^21]

多年来出现的一个重要概念是递归，简单来说，这意味着一个证明可以用来证明另一个证明的正确性。本文中提出的场景涉及一个证明者希望证明一个可能很长的计算结果的正确性。给定一个图灵机，我们可以证明状态转移函数的单个步骤的正确性，但这还不够；我们想证明整个计算的正确性，这由一系列状态转移组成。

增量可验证计算（IVC）背后的想法如下：假设我们可以证明从 S1 到 S2 的单个状态转移是正确的。然后，我们可以将两个证明合并为一个：证明者表明他们知道两个有效证明：



合并的证明将说服验证者从 S1 到 S3 的转移是正确的。这个过程可以对任意数量的步骤重复，使我们能够将任意长的计算压缩为一个证明（更具体地说，是多项式长的计算）。

重要的是要注意，这个构造依赖于两个关键假设：

尽管这个构造在理论上是强大的，但在实践中应用成本高。为了解决这个问题，提出了新的方法来提高效率。其中之一是使用折叠方案 [^22] ，它放宽了假设并避免了递归 SNARK 验证的需要。折叠的想法是，给定两个证明，π 和 π′，我们可以将它们“折叠”成一个单一的证明，π″。验证者相信，如果折叠实例是可满足的，则原始实例也是可满足的。

## 10 - 通过 zkVM 进行可验证计算

Ben-Sasson, Chiesa, Tromer, Virza - 适用于冯·诺依曼架构的简洁非交互式零知识 (2014) [^23]

这篇最终的论文讨论了第一个实用的 零知识虚拟机 (zkVM) 构造，这是一种能够执行任意程序并生成这些计算正确性证明的虚拟机。所描述的机器遵循冯·诺依曼架构，这意味着程序和数据存储在同一内存中。大多数现代 CPU 基于这一范式，因此，从理论上讲，任何可以在经典计算机上运行的程序也可以在该架构上运行。

论文介绍了一种称为 vnTinyRAM 的 RISC 架构，并展示了一个移植到该指令集的 C 编译器。证明系统旨在验证程序执行的正确性，直到固定的步骤数为止。其基本思想是电路被构造为一个重复的状态转换函数，展开直到达到指令计数限制。

如今，zkVM 越来越受欢迎。它们的一个关键优势是用户可以使用 高级编程语言 编写程序并用它们生成证明。这相较于手动编写电路提供了显著的优势，因为许多标准算法和数据结构已经在这些高级语言中定义。此外，开发者可以重用熟悉的计算模型，这大大降低了使用零知识证明的学习曲线。

还值得注意的是，许多 zk-rollup 基于这一模型。例如，支持以太坊虚拟机 (EVM) 执行的 zk-rollup 使用 zkVM 来证明 EVM 执行的正确性。

最后，论文介绍了其自身的架构，优化用于零知识证明系统。另一个流行的 zk 友好架构示例是 Cairo CPU 架构 [^24] ，这是一个经过优化以使用 STARKs 进行证明的图灵完备 CPU。

## 参考论文

[^1]: Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. (link) ↩︎

[^2]: Fiat, A., & Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. (link) ↩︎[^3]: Goldreich, O., Micali, S., & Wigderson, A. (1987). How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design. (link) ↩︎[^4]: Ben-Or, M., Goldreich, O., Goldwasser, S., Håstad, J., Kilian, J., Micali, S., & Rogaway, P. (1990). Everything provable is provable in zero-knowledge. (link) ↩︎[^5]: Shamir, A. (1992). IP=PSPACE. (link) ↩︎[^6]: Micali, S. (2000). Computationally sound proofs. ([link](https://people.csail.mit.edu/silvio/Selected Scientific Papers/Proof Systems/Computationally\_Sound\_Proofs.pdf)) ↩︎[^7]: Cook, S. A. (1971). Proof Verification and Hardness of Approximation Problems. (link) ↩︎[^8]: Kilian, J. (1992, July). A note on efficient zero-knowledge proofs and arguments. (link) ↩︎[^9]: Goldwasser, S., Kalai, Y. T., & Rothblum, G. N. (2015). Delegating computation: interactive proofs for muggles. (link, see also this note by Justin Thaler) ↩︎[^10]: Lund, C., Fortnow, L., Karloff, H., & Nisan, N. (1992). Algebraic methods for interactive proof systems. (link) ↩︎[^11]: Thaler, J. (2015). A note on the GKR protocol. (link) ↩︎[^12]: Gennaro, R., Gentry, C., Parno, B., & Raykova, M. (2013). Quadratic span programs and succinct NIZKs without PCPs. (link) ↩︎[^13]: Groth, J. (2010). Short pairing-based non-interactive zero-knowledge arguments. (link) ↩︎[^14]: Parno, B., Howell, J., Gentry, C., & Raykova, M. (2016). Pinocchio: Nearly practical verifiable computation. (link) ↩︎[^15]: Groth, J. (2016). On the size of pairing-based non-interactive arguments. (link) ↩︎[^16]: Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). Zerocash: Decentralized anonymous payments from bitcoin. (link) ↩︎[^17]: Gabizon, A., Williamson, Z. J., & Ciobotaru, O. (2019). Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. (link) ↩︎[^18]: Kate, A., Zaverucha, G. M., & Goldberg, I. (2010). Constant-size commitments to polynomials and their applications. (link) ↩︎[^19]: Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2018). Scalable, transparent, and post-quantum secure computational integrity. (link) ↩︎[^20]: Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2018). Fast reed-solomon interactive oracle proofs of proximity. (link) ↩︎[^21]: Valiant, P. (2008). Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. (link) ↩︎[^22]: Kothapalli, A., Setty, S., & Tzialla, I. (2022, August). Nova: Recursive zero-knowledge arguments from folding schemes. (link) ↩︎[^23]: Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014). Succinct Non-Interactive zero knowledge for a von neumann architecture. (link) ↩︎[^24]: Goldberg, L., Papini, S., & Riabzev, M. (2021). Cairo–a Turing-complete STARK-friendly CPU architecture. (link) ↩︎