ZK技术的历史发展脉络梳理

零知识证明(ZK Proofs)是功能强大的密码学原语，允许一方（证明者）在不透露任何私密信息的情况下，使另一方（验证者）相信某个给定的声明是真实有效的。近年来ZK在可验证私密计算、为计算机程序提供有效性证明以及区块链领域获得了广泛关注，并且对世界的发展产生了重大的积极作用。

虽然ZK是新兴技术，但其基本思想和概念可以追溯到上世纪80年代。在与比特币和以太坊等区块链结合后，ZK技术的发展显著加速，因为区块链可以通过SNARK和STARK进行有效性证明，极大程度的增强可扩展性，这使ZK在区块链领域中炙手可热。

正如Starkware创始人Eli Ben-Sasson所言，近年来我们见证了密码学证明系统的“寒武纪大爆发”，每种证明系统各有独特的优势和劣势，并且在设计时进行了权衡。硬件的进步、更好的算法、新的论点和周边工具，都刺激了ZK系统的性能提升及新式系统的诞生。许多证明系已经在实际应用中被采用，而人们仍在不断扩展ZK的边界。

这也促使人们深入思考一个问题：是否有一个适用于所有应用的通用ZK证明系统？对此我们认为这种可能性不大，原因有三点：

1. 应用程序的多样性；

2. 不同的约束类型（包括内存、验证时间、证明时间）；

3. 对鲁棒性的需求（如果一种证明系统被黑客攻破，我们仍然可以切换到其他系统作为保险）。

基于上述理由，ZK证明系统理应是多样性的。但即使证明系统的种类很多，也一定有一个显著的共性：ZK证明可以被快速的验证，并且拥有一个验证层，可以很容易地适应新的证明系统，以解决其依附的基础层（如以太坊）的相关困难。

在ZK领域中，zk-SNARK被频繁提及。它是实现零知识证明的一种形式，通过使用复杂的数学工具，如双线性配对和算术电路，来实现高效的零知识证明。zk-SNARK的特点是证明过程简洁化、非交互式，证明者和验证者之间只需要单次通讯不需要多次交互。此外，zk-SNARK的证明尺寸非常短小，验证效率高，适合在资源有限的环境中使用。

而zk-STARK是另一种常见的形式，旨在克服zk-SNARK的某些局限性。zk-STARK不依赖于可信设置，使用更透明的数学构造系统，如多项式承诺和有限域运算、哈希碰撞等，来生成和验证证明。zk-STARK比zk-SNARK更具可扩展性，适用于更大规模的计算，证明生成速度更快，但是Proof本身的尺寸通常较大。

可以说，zk-SNARK和zk-STARK都是零知识证明中常用的形式，但它们在透明度、可扩展性、证明大小等方面有所不同。

总体来看，一个ZK证明系统通常包括PIOP（多项式交互式预言机）和PCS（多项式承诺方案）两大部分。常见的PIOP方案包括PLONKish、GKR等，而常见的PCS方案包括FRI，KZG，IPA等，比如Zcash版本的Halo2使用了Plonkish+IPA的实现方式，至于zk-STARK其实可以当成是一种基于FRI的特殊的zk-SNARK。

如果更详细的说，不同类型的证明系统会使用不同的多项式承诺方案(PCS)、算术化方案、交互式预言机证明(IOP)或概率可检查证明(PCP)。

进一步说，不同的ZK证明系统往往在如下指标上有所不同：

密码学假设：抗碰撞哈希函数、椭圆曲线上的离散对数问题、指数知识

透明设置vs可信设置

生成证明的耗时：线性vs超线性

验证证明的耗时：常数时间、对数时间、次线性、线性

证明尺寸的大小

递归的简易性

算术化方案

单变量vs多变量多项式

下文中我们将简要谈及ZK技术的起源，探索其基本的构建模块，概述不同ZK证明系统的兴起和衰落过程。同时，本文并不对证明系统本身进行详尽分析，而是着重介绍那些对该领域产生深远影响的人，毕竟任何行业的发展只有通过先驱者的伟大想法并诉诸实践，才有可能实现。

zk-SNARK的历史发展脉络

起源：20世纪80~90年代

正如我们所提到的，零知识证明并不是新概念，其定义、基础、重要定理，甚至相关的重要协议，早在上世纪80年代中期就已经出现，首次出现是在是在Goldwasser、Micali(Algorand创始人)和Rackoff的论文《The Knowledge Complexity of Interactive Proof Systems》中。

而如今我们用来构建ZK-SNARK技术的关键思想和协议，在20世纪90年代就被出，比如Sumcheck协议，将对多元多项式求值总和的声明，简化为在椭圆曲线上随机选择的点进行单一求值，该协议为ZK技术奠定了重要基础。

所以，ZK思想的萌芽实际上远远早于比特币的出现，但在当时普遍缺乏ZK的合适用例，人们也无法提供满足ZK证明系统所需的强大算力，毕竟互联网和硬件设备在上世纪90年代并不发达。

GKR协议（2007)

GKR（Goldwasser-Kalai-Rothblum）是一种交互式协议，证明者的运行时间与电路中逻辑门的数量呈线性相关，而验证者的耗时则与电路大小呈次线性关系。在GKR协议中，证明者和验证者需要对一个有限域上的双输入算术电路运行结果达成一致，该电路的深度为d，第d层为输入层，第0层为输出层。协议从关于电路输出的声明开始，通过递归将其简化为对上一层的声明。最后，我们可以将对输出的声明转换为对电路输入参数的声明，这很容易被验证。可以说，GKR协议是在前面提及的Sumcheck协议基础上进行了高度简化的。

KZG多项式承诺方案(2010)

2010年，三名ZK领域的专家——来自德国研究机构MPI-SWS的Kate，来自加拿大密码学公司Certicom Research的Zaverucha，以及来自滑铁卢大学的Goldberg联合发表了一篇论文《Constant-Size Commitments to Polynomialsand Their Applications》。该论文提出了一种使用双线性对群的多项式承诺方案，名为KZG。

该承诺由一个单独的群元素组成，提交者可以高效地揭示多项式的任何正确求值，借助批处理技术，可以对多个多项式的求值进行揭示。KZG承诺成为了一些知名ZK证明系统的基本构建模块之一（比如以太坊PES小组用的halo2），更是在以太坊的EIP-4844中起到了核心作用。若要更直观地了解批处理技术的概念，可以参考关于Mina-Ethereum桥的文章Mina-Ethereum bridge。

参考资料：https://blog.lambdaclass.com/mina-to-ethereum-bridge/

基于椭圆曲线的实用ZK-SNARK系统（2013）

ZK-SNARK的第一个实用结构出现在2013年，需要一个预处理步骤来生成证明密钥和验证密钥，并且是随程序或电路特定的，没有泛用化。这些密钥的尺寸可能非常大，并取决于秘密参数本身；若这种保密性被破坏，攻击者就可以伪造出证明。在这种实用的ZK-SNARK系统中，要将代码转换为可以被证明的形式，需要将代码编译为一组数学形式的多项式约束。

起初，上述过程必须手动完成，既耗时又容易出错。后来针对该方向的技术更迭，主要试图解决下述核心问题：

提供更高效的证明

减少预处理的次数

实现通用的而非电路特定的设置

避免可信设置

开发使用高级语言描述电路的方法，而不是手动编写多项式约束

Pinocchio协议（2013）

Pinocchio协议是第一个实际可用的zk-SNARK系统，基于二次算术程序（QAP），最初的证明大小为288字节。Pinocchio的工具链提供了一个将C语言编译为算术电路的编译器，它可以进一步转换为QAP。Pinocchio协议要求验证者生成密钥，这些密钥并不通用，而是由电路特定的。该证明系统生成和密钥设置的渐进时间复杂度与计算规模呈线性关系，验证时间与公共输入和输出的大小呈线性关系。

Groth16(2016)

Groth引入了一种新的ZK明算法，在处理R1CS上具有更高的性能。R1CS即Rank-1 Con­straint Sys­tem，一阶约束系统，是zk-SNARK中的一种多项式约束形式。Gorth的证明是数据规模最小的（仅包含三个群元素），且验证速度很快，只需进行三个配对运算，以及一个使参考字符串结构化的预处理步骤。但Gorth主要的缺点是每个需要证明的程序都需要进行不同的可信设置，这在实际应用中相当不便。

后来Groth16被用于ZCash，后者是一个比较有名的隐私区块链项目（Starkware创始人Eli参与做的）。

Bulletproofs与IPA（2016）

前面提到的KZG多项式承诺方案，其一大弱点是需要可信设置。Bootle等人提出了一种有效的零知识证明系统，该系统对满足内在乘积关系的Pedersen承诺的开启进行了分析。内积证明具有线性复杂度的证明耗时，证明者和验证者之间的交互次数是对数级的，但验证时间是线性的。此外Bootle等人还开发了一种不需要可信设置的多项式承诺方案。这些思想后来被Halo2和Kimchi等采用。

Sonic、Marlin和Plonk（2019）

Sonic、Plonk和Marlin解决了Groth16算法中每个程序都需要可信设置的问题，引入了通用且可更新的结构化参考字符串（用来实现仅需一次的可信设置）。其中，Marlin提供了一个基于R1CS的证明系统，并且成为了Aleo的核心技术。

而Plonk引入了一种新的算术方案（后来被称为Plonkish）以及使用grand-product来检查复制约束。Plonkish还允许引入用于特定操作的专用电路逻辑门，即所谓的“自定义门”。许多知名的区块链项目方都用到了Plonk的定制化版本，包括Aztec、zkSync、Polygon zkEVM、Mina、以太坊PSE小组和Scroll等。

Spartan（2019）

Spartan为使用R1CS描述的电路提供了一个IOP，利用了多变量多项式和求和检验协议的特性。通过使用合适的多项式承诺方案，它实现了一套具有透明性的zk-SNARK系统，并且生成证明的时间复杂度是线性的。

Lookups(2020)

Gabizon和Williamson于2020年在论文中提出了plookup，利用grand-product证明某个值包含在预先计算出的真值表中，展示了如何将plookup参数引入Plonk算法。

然而，这些lookup arguments有一个共同的问题，证明者需要耗费巨大成本建立完整的真值表，因此之前围绕着Lookups的工作都致力于将证明成本减少。

后来Haböck在论文中引入了LogUp，它使用对数导数将grand-product检查转化为倒数之和。LogUp对于Polygon zkEVM的性能提升至关重要，因为他们需要将整个真值表拆分为多个STARK模块。这些模块必须正确链接，而跨表查找可以强制实现这一点。此后LogUp-GKR的引入又通过GKR协议提高了LogUp的性能。

Caulk是第一个使证明时间与真值表大小呈亚线性关系的方案，它的预处理时间复杂度为O(NlogN)，存储占用的空间复杂度为O(N)，其中N是真值表大小。随后又出现了其他方案，如Baloo、flookup、cq和caulk+。此外，Lasso提出了若干改进方案，避免在真值表具有特定结构时对其进行承诺。

HyperPlonk（2022）

HyperPlonk在论文《HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates》中被提出。HyperPlonk基于Plonk的理念，采用多变量多项式。它不使用除法来检查约束的执行，而是依赖于求和检验协议。同时，它还支持高阶约束，而不会影响证明生成的时间。

由于使用了多变量多项式，无需执行快速傅里叶变换（FFT），证明生成的时间与电路规模成线性关系。HyperPlonk还引入了一种适用于小字段的新置换IOP，并且采用基于求和检验的协议，减少了证明者的工作量、证明大小，以及验证时间。

使用防碰撞哈希函数的ZK证明系统

在2013年Pinocchio被提出的同时，有一些关于生成电路/算术化方案的方案，这些方案可以证明虚拟机对指令的执行结果正确。尽管为虚拟机开发算术化方案比为某些程序编写专用电路更复杂或效率更低，但它却有一个重要优势：无论程序多复杂，只需证明其在虚拟机中是正确执行的即可。

TinyRAM中的一些想法后来在Cairo虚拟机的设计中得到了改进，随后又有了zk-evm和通用zkvm等。在证明系统中使用抗碰撞的哈希函数消除了对可信设置或椭圆曲线操作的需求，但代价是证明时间更长。

TinyRAM（2013）

在“SNARKs for C”中，他们基于PCP开发了一种证明系统，用于证明C语言编写的程序的执行结果正确。该程序被编译为TinyRAM，一种简化的VM。该VM具有字节级可寻址的随机存储器，电路大小在计算规模上呈准线性增长，可以高效地处理循环、控制流和内存访问等操作。

其中，PCP指Probabilistically Checkable Proof，即概率可检查证明，验证者只需阅读证明中随机选择的一小部分内容，就能以很高的置信度检查证明的有效性。与验证者需要检查整个证明的传统证明系统不同，PCP只需有限的随机性即可实现高效验证。

Ligero（2017）

Ligero引入了一套证明系统，该系统可实现大小为O(√￣n)的证明，其中n是电路的大小。它以矩阵形式排列多项式系数。Brakedown基于Ligero构建，并引入了领域无关的多项式承诺方案的概念。

STARKs（2018）

STARKs（Scalable Transparent ARguments of Knowledge）由Eli Ben-Sasson等人于2018年提出。它们实现了?(log²⁡?)的证明复杂度，具有快速的验证速度，不需要可信设置，并且被推测为后量子安全。它们被Starkware/Starknet与Cairo虚拟机一起投入采用。其关键创新包括代数中间表示（AIR）和快速Reed-Solomon交互式Oracle接近证明（FRI）协议。另外，STARKs也被许多知名的区块链项目所使用（如Polygon Miden、RiscZero、Winterfell、Neptune以及ZeroSync、zkSync等）。

新的发展方向

不同的证明系统在实际应用中的使用展示了不同方法的优点，并推动了ZK的发展。例如，Plonkish的算术化方案提供了一种简单的方法，来包含自定义逻辑门和lookup arguments；FRI已经显示出作为PCS的出色性能，促成了Plonky的诞生。同时，在AIR中使用grand-products检查（带来了预处理的随机化AIR）提高了其性能并简化了内存访问参数。zk-STARK由于在生成效率上更好，且有越来越多的ZK友好型哈希函数被引入，而越来越受欢迎。

新的多项式承诺方案（2023）

随着基于多变量多项式的高效SNARK（如Spartan或HyperPlonk）的出现，人们对适用于此类多项式的新承诺方案的兴趣日益增加。Binius、Zeromorph和Basefold都提出了新的方式来承诺多线性多项式。Binius的优势在于表示数据类型时没有额外开销（而许多其他证明系统至少使用32位字段元素来表示单个位），并且在二进制域上工作。该承诺方案采用了为领域无关而设计的brakedown。Basefold将FRI推广到除Reed-Solomon之外，从而实现了领域无关的多项式承诺方案（PCS）。

领域无关是多项式承诺方案的一个性质，指多项式承诺方案中，承诺过程不依赖于任何特定领域的特定属性。这意味着可以对任何代数结构的多项式做出承诺，如有限域、椭圆曲线，甚至整数环。

可定制约束系统（2023）

CCS泛化了R1CS，同时捕捉了R1CS、Plonkish和AIR的算术化，而没有额外开销。使用CCS与Spartan IOP结合可以产生SuperSpartan，它支持高维度约束，而证明者无需承担与约束阶数成比例的加密成本。特别地，SuperSpartan为AIR提供了一个具有线性时间证明的SNARK。

总结

这篇文章综述了自上世纪80年代中期以来ZK技术的进展。计算机科学、数学和硬件的进步，加上区块链的引入，催生了新的、更高效的ZK证明系统出现，为许多可能改变社会的应用开辟了道路。

研究人员和工程师们根据需求提出了ZK系统的改进方案，重点围绕在证明尺寸大小、内存使用程度、透明度、抗量子安全性、证明时间和验证时间等方面。虽然一直以来，ZK的主流实现方案有两大类（SNARK与STARKs），但这两者之间的界限已经逐渐模糊，不同证明系统的优势正被结合起来，例如结合不同的算术化方案与新的多项式承诺方案。

我们可以预期，新的ZK证明系统将继续涌现，且性能会不断提升。对于使用这些证明系统的应用来说，如果不能跟随最新技术的迭代发展，不断重构并应用最新的算法，现在的领先地位也只是暂时的。

原文链接：https://blog.lambdaclass.com/our-highly-subjective-view-on-the-history-of-zero-knowledge-proofs/