证明、论证与零知识

Kurt Pan

2023 年 5 月 10 日

目录

Chapter 1

引言

本书讨论了可验证计算（VC）。可验证计算是指一种名为交互式证明（IPs）和论证的密码学协议，证明者能够向验证者提供证明者正确地执行了所请求的计算的保证。交互式证明和论证在 20 世纪 80 年代被提出，代表了一个陈述为真的"证明"包括什么概念上的重大扩展。传统上的证明是一个静态对象，可以很容易地通过逐步检查来检验正确性，因为证明的每个单独步骤都应该易于验证。相比之下，交互式证明允许证明者和验证者之间的进行交互，以及允许无效证明以微小但非零的概率通过验证。交互式证明和论证之间的区别在于，论证（而非交互式证明）允许存在对错误陈述的"证明"，只要找到这些"证明"需要巨大的计算能力就可以。[1]

20 世纪 80 年代中期和 90 年代初的著名理论结果表明，至少原则上讲，可验证计算协议可以达到惊人的成就：包括让手机可以去监控强大但不受信任（甚至恶意）的超级计算机的执行，让计算能力较弱的外设（例如，安全卡读卡器）将安全核心工作外包给强大的远程服务器，以及让数学家仅通过查看所谓证明中的几个符号就能对定理的正确性具有很高的信心。[2]

当 VC 协议具有一种称为*零知识*的性质时，它会在密码学环境中非常有用。零知识的意思是证明或论证除了其本身的有效性之外，不会泄露任何信息。

为了具体说明为什么零知识协议有用，考虑以下来自身份认证的典型例子。假如 Alice 选择了一个随机口令 x，公开了一个哈希值 $z = h(x)$。其中 h 是一个单向函数，给定一个对随机选择 x 的 $z = h(x)$，需要大量计算能力才能找到 z 在 h 下的原像，即一个满足 $h(x') = z$ 的 x'。假如 Alice 在之后想要说服 Bob 她是发布 z 的同一个人。她可以通过向 Bob 证明她知道一个满足 $h(x') = z$ 的 x' 来实现这一点。这将使 Bob 相信 Alice 是发布 z 的同一个人，因为这意味着 Alice 要么一开始就知道 x，要么她反转了 h （这被认为是超出了 Alice 的计算能力）。

Alice 如何让 Bob 相信她知道 h 下 z 的原像？一个简单的证明是 Alice 将 x 发送给 Bob，Bob 可

[1] 比如说，一个不是 IP 的论证系统，可能会使用密码系统，使得作弊的证明者有可能找到一个可通过验证的对假陈述的"证明"，当且仅当证明者可以攻破密码系统。

[2] 只要证明以一种特定的、略微有些冗余的形式书写。具体见第9章中对概率可检验证明（PCPs）的讨论。

以轻松检验 $h(x)=z$。但这样所揭示的信息比 Alice 知道 z 的原像要多得多。具体来说它揭示了原像本身。Bob 可以使用该知识永远冒充 Alice，因为现在他也知道 z 的原像了。

为了防止 Bob 获悉可能泄露口令 x 的信息，给出不能揭示任何超出其自身有效性的内容的证明是很重要的。这正是零知识性质所保证的。

本书的一个特定目标是描述构造所谓的零知识简洁非交互式知识论证（简称 zk-SNARKs）的各种方法。“简洁”意味着证明很短。“非交互”意味着证明是静态的，由来自证明者的单个消息组成。（第二个）“知识”大致意味着协议不仅确定一个陈述为真，而且证明者*知晓*一个“证据”来证明该陈述为真。[3]满足所有这些性质的论证系统在整个密码学中有无数应用。

对密码学相关的高度特定化的陈述（如证明离散对数的知识 [Sch89]）的实用零知识协议已经有了数十年了。然而，通用零知识协议直到最近才变得足够高效，可以用于密码部署中。通用的意思是，协议设计技术适用于任意计算。这一令人兴奋的进展包括了漂亮的新协议的提出，并引发了各界对零知识证明和论证的浓厚兴趣。本书旨在以统一的方式让人们能够轻松理解这些协议设计的主要思想和方法。

背景和上下文 在 20 世纪 80 年代中期和 90 年代，理论计算机科学家理论计算机科学家发现 IP 和论证系统可以比传统的静态和信息论安全 [4]的 **NP** 证明（至少在渐近意义上）更高效。刻画这些协议能力的基础定理（例如 **IP** = **PSPACE** [LFK+92; Sha92]、**MIP** = **NEXP** [BFL91] 和 PCP 定理 [ALM+98; AS98]）都是计算复杂性理论中最具影响力和最著名的定理。[5]

零知识协议设计方法及本书的理念 论证系统通常分两步开发。开发一个信息论安全协议（例如 IP、*多证明者交互式证明* (MIP) 或*概率可检查证明* (PCP)），以对单个或多个证明者假定以某种受限方式行事建模。（例如，在 MIP 中，会假定证明者不会相互发送有关他们从验证者那里收到的挑战的信息）。第二步，信息论安全的协议与密码学相结合，以“强制”（单个）证明者以受限方式行事，从而产生一个论证系统。这第二步通常还会赋予所生成的论证系统重要的性质，例如零知识、简洁性和非交互性。如果生成的论证系统满足所有这些性质，那么它实际上是一个 zk-SNARK。

本书中信息论安全模型的顺序 我们首先介绍 IP，然后是 MIP，然后是 PCP 和 IOP，再然后是线性 PCP。该顺序大致遵循研究文献中模型出现的时间顺序。有点讽刺的是，这些模型是以类似于*倒*时间序的方式应用于实际的 SNARK 设计中的。比如说第一个实用的 SNARK 就是基于线性 PCP 的。

[3] 例如，上述身份验证场景中确实需要对“存在一个口令 x 使得 $h(x)=z$”这一陈述进行零知识*知识*证明。这是因为应用要求 Bob 不仅要确信在 h 下*存在*z 的原像 x（如果 h 是满射函数，这将始终为真），而且还要确信 Alice 知道 x。

[4] 信息论安全

[5] The results IP=PSPACE and MIP=NEXP are both covered in this survey (see Sections 4.5.5 and 8.5 respectively).

事实上这并非巧合：引入线性 PCP 的主要动机就是为了得到更简单和更实用的简洁论证，特别是考虑到从 PCP 导出的论证的*实际不可用性*。

逐章大纲

阅读本书的建议

首次阅读可跳过的材料 Sections 4.2-4.5 are devoted to detailed example applications of the sum-check protocol and explaining how to efficiently implement the prover within it. While these sections contain interesting results and are useful for familiarizing oneself with the sum-check proto- col, subsequent chapters do not depend on them. Similarly, Chapter 5 on the Fiat-Shamir transformation and Chapter 6 on front-ends are optional on a first reading. Sections 9.3 and 9.4 provide PCPs that are mainly of historical interest and can be skipped. Chapters 11 and 13 offer treatments of zero-knowledge that largely stand on their own. Similarly, Chapter 18 discusses SNARK composition and stands on its own.

1.1 数学证明

本书涵盖了*数学证明*的不同概念及其在计算机科学和密码学中的应用。非正式地，我们所说的证明是指任何能使某人相信某项陈述为真的东西，而“证明系统”是任何决定什么是令人信服的证明以及什么不是的程序。也就是说，证明系统由验证程序决定，该验证程序将任何陈述和声称该陈述为真的“证明”作为输入，并判定证明是否有效。

我们希望证明系统具有哪些性质呢？这有四个比较明显要有的。

- 任何真陈述都应该有其有效性的令人信服的证明。此性质通常称为*完整性*。
- 任何假陈述都不应有令人信服的证明。此性质称为*可靠性*。
- 理想情况下，验证程序将是“高效的”。粗略地说，这意味着简单的陈述应该有简短的（令人信服的）证明，且可以快速*检验*。
- 理想情况下，证明生成也应该是高效的。粗略地说，这意味着简单的陈述应该有可以快速*找到*的简短（令人信服的）证明。

传统上，数学证明是可以逐行编写并检查其正确性的东西。这种传统的证明概念正是复杂性类 **NP**[8] 所捕获的概念。但是，[8]，在过去的 30+ 年里，计算机科学家研究了更普遍和奇异的证明概念. 这改变了计算机科学家对证明某事意味着什么的观念，并导致了复杂性理论和密码学的重大进步。

This survey covers different notions of mathematical proofs and their applications in computer science and cryptography. Informally, what we mean by a proof is anything that convinces someone that a statement is true, and a ”proof system” is any procedure that decides what is and is not a convincing proof. That is, a proof system is specified by a verification procedure that takes as input any statement and a claimed ”proof” that the statement is true, and decides whether or not the proof is valid. What properties do we want in a proof system? Here are four obvious ones. - Any true statement should have a convincing proof of its validity. This property is typically referred to as completeness. - No false statement should have a convincing proof. This property is referred to as soundness. - Ideally, the verification procedure will be ”efficient”. Roughly, this means that simple statements should have short (convincing) proofs that can be checked quickly. - Ideally, proving should be efficient too. Roughly, this means that simple statements should have short (convincing) proofs that can be found quickly.

Traditionally, a mathematical proof is something that can be written and checked line-by-line for correctness. This traditional notion of proof is precisely the one captured by the complexity class **NP**[8] However, [8], over the last 30+ years, computer scientists have studied much more general and exotic notions of proofs. This has transformed computer scientists’ notions of what it means to prove something, and has led to major advances in complexity theory and cryptography.

1.2 我们将研究哪些非传统证明？

本书中研究的所有证明概念本质上都是概率性的。即验证程序将做出随机选择，且可靠性将以（非常）高的概率在这些随机选择上成立。也就是说，验证程序会以（非常）小的概率宣布假陈述为真。

1.2.1 交互式证明（IP）

To understand what an interactive proof is, it is helpful to think of the following application. Imagine a business (verifier) that is using a commercial cloud computing provider to store and process its data. The business sends all of its data up to the cloud (prover), which stores it, while the business stores only a very small “secret”summary of the data (meaning that the cloud does not know the user’s secret summary). Later, the business asks the cloud a question about its data, typically in the form of a computer program f that the business wants the cloud to run on its data using the cloud’s vast computing infrastructure. The cloud does so, and sends the user the claimed output of the program, f (data). Rather than blindly trust that the cloud executed the program on the data correctly, the business can use an interactive proof system (IP) to obtain a formal guarantee that the claimed output is correct. In the IP, the business interrogates the cloud, sending a sequence of challenges and receiving a sequence of responses. At the end of the interrogation, the business must decide whether to accept the answer as valid or reject it as invalid. See Figure 1.1 for a diagram of this interaction. Completeness of the IP means that if the cloud correctly runs the program on the data and follows the prescribed protocol, then the user will be convinced to accept the answer as valid. Soundness of the IP means that if the cloud returns the wrong output, then the user will reject the answer as invalid with high probability no matter how hard the cloud works to trick the user into accepting the answer as valid. Intuitively, the interactive nature of the IP lets the business exploit the element of surprise (i.e., the fact that the cloud cannot predict the business's next challenge) to catch a lying cloud in a lie. It is worth remarking on an interesting difference between IPs and traditional static proofs. Static proofs are transferrable, meaning that if Peggy (prover) hands Victor (verifier) a proof that a statement is true, Victor can turn around and convince Tammy (a third party) that the same statement is true, simply by copying the proof. In contrast, an interactive proof may not be transferrable. Victor can try to convince Tammy that the statement is true by sending Tammy a transcript of his interaction with Peggy, but Tammy will not be convinced unless Tammy trusts that Victor correctly represented the interaction. This is because soundness of the IP only holds if, every time Peggy sends a response to Victor, Peggy does not know what challenge Victor will respond with next. The transcript alone does not give Tammy a guarantee that this holds.

1.2.2 论证系统

论证系统就是 IP，只是其中的可靠性性要求只需要防止在多项式时间内运行的作弊证明者。[6] 论证系统会使用密码学。粗略地说，在论证系统中，作弊证明者无法欺骗验证者接受假陈述，除非它攻破了某个密码系统，且破解该密码系统被认为需要超多项式时间。

1.2.3 多证明者交互式证明（MIP）/概率可检验证明（PCP）等

MIP 类似于 IP，但有多个证明者，并且假设这些证明者不会相互共享他们从验证者那里收到的挑战的信息。MIPs 的一个常见类比是在审讯之前将两名或更多犯罪嫌疑人放在不同的房间里，看他们是否能讲出一致的故事。执法人员可能不会对 MIPs 研究为这种做法提供了理论依据感到意外。具体来说，MIPs 的研究表明，将证明者锁在不同的房间里，然后分别审问他们，这可以说服审问者相信比在一起审问时更复杂的陈述。

[6] 这大致意味着如果输入的大小为 n，那么证明者的运行时间（对于足够大的 n 值）应该以 n 的某个常数幂为界，例如 n^{10}。

In a PCP, the proof is static as in a traditional mathematical proof, but the verifier is only allowed to read a small number of (possibly randomly chosen) characters from the proof. [10] This is in analogy to a lazy referee for a mathematical journal, who does not feel like painstakingly checking the proofs in a submitted paper for correctness. The PCP theorem [ALM$^+$98, AS98] essentially states that any traditional mathematical proof can be written in a format that enables this lazy reviewer to obtain a high degree of confidence in the validity of the proof by inspecting just a few words of it.

在 PCP 中，证明与传统数学证明一样是静态的，但验证者只能从证明中读取少量（可能是随机选择的）字符。[7] 这类似于数学期刊的懒惰审稿人，他不想煞费苦心地检查提交论文中的证明是否正确。PCP 定理 [ALM$^+$98, AS98] 本质上指出，任何传统的数学证明都可以用一种格式编写，使这个懒惰的审阅者能够对证明的有效性获得高度的信心通过检查它的几个词。

Philosophically, MIPs and PCPs are extremely interesting objects to study, but they are not directly applicable in most cryptographic settings, because they make unrealistic or onerous assumptions about the prover(s). For example, soundness of any MIP only holds if the provers do not share information with each other regarding what challenges they receive from the verifier. This is not directly useful in most cryptographic settings, because typically in these settings there is only a single prover, and even if there is more than one, there is no way to force the provers not to communicate. Similarly, although the verifier only reads a few characters of a PCP, a direct implementation of a PCP would require the prover to transmit the whole proof to the verifier, and this would be the dominant cost in most real-world scenarios (the example of a lazy journal referee notwithstanding). That is, once the prover transmits the whole proof to the verifier, there is little real-world benefit to having the verifier avoid reading the whole proof.

However, by combining MIPs and PCPs with cryptography, we will see how to turn them into argument systems, and these are directly applicable in cryptographic settings. For example, we will see in Section 9.2 how to turn a PCP into an argument system in which the prover does not have to send the whole PCP to the verifier.

Section 10.2 of this survey in fact provides a unifying abstraction, called polynomial IOPs, of which all of the IPs, MIPs, and PCPs that we cover are a special case. It turns out that any polynomial IOP can be transformed into an argument system with short proofs, via a cryptographic primitive called a polynomial commitment scheme.

[7] More precisely, a PCP verifier is allowed to read as much of the proof as it wants. However, for the PCP to be considered efficient, it must be the case that the verifier only needs to read a tiny fraction of the proof to ascertain with high confidence whether or not the proof is valid.

Chapter 2

随机性的力量：指纹和 Freivalds 算法

Chapter 3

定义和技术预备知识

Chapter 4

交互式证明系统

Chapter 5

使用 Fiat-Shamir 得到公开可验证非交互式论证

Chapter 6

前端：将计算机程序转化为电路

Chapter 7

基于交互式证明的首个电路可满足性简洁论证

Chapter 8

MIP 和简洁论证系统

Chapter 9

PCP 和简洁论证系统

Chapter 10

交互式预言证明

Chapter 11

零知识证明和论证系统

Chapter 12

基于离散对数困难性的 Σ-协议和承诺

Chapter 13

通过承诺并证明和遮盖多项式实现零知识

Chapter 14

基于离散对数困难性的多项式承诺

Chapter 15

基于配对的多项式承诺

Chapter 16

多项式承诺总结

Chapter 17

线性 PCP 和简洁论证

Chapter 18

SNARK 组合和递归

Chapter 19

实用论证系统鸟瞰

Bibliography

[ALM+98] Sanjeev Arora et al. *Proof verification and the hardness of approximation problems.* Journal of the ACM (JACM) 45.3 (1998), pp. 501–555 (Cited on page 6).

[AS98] Sanjeev Arora and Shmuel Safra. *Probabilistic checking of proofs: A new characterization of NP.* Journal of the ACM (JACM) 45.1 (1998), pp. 70–122 (Cited on page 6).

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. *Non-deterministic exponential time has two-prover interactive protocols.* Computational complexity 1 (1991), pp. 3–40 (Cited on page 6).

[LFK+92] Carsten Lund et al. *Algebraic methods for interactive proof systems.* Journal of the ACM (JACM) 39.4 (1992), pp. 859–868 (Cited on page 6).

[Sch89] Claus-Peter Schnorr. *Efficient Identification and Signatures for Smart Cards.* Annual International Cryptology Conference. 1989 (Cited on page 6).

[Sha92] Adi Shamir. *IP = PSPACE.* Journal of the ACM (JACM) 39.4 (1992), pp. 869–877 (Cited on page 6).