The Internet Computer for Geeks
极客的互联网电脑

(v1.0)
(v1.0)

The DFINITY Team*

明确小组 *

January 21, 2022
2022 年 1 月 21 日

Abstract
摘要

Smart contracts are a new form of software that will revolutionize how software is written, IT systems are maintained, and applications and whole businesses are built. Smart contracts are composable and autonomous pieces of software that run on decentralized blockchains, which makes them tamperproof and unstoppable. In this paper, we describe the Internet Computer (IC), which is a radical new design of blockchain that unleashes the full potential of smart contracts, overcoming the limitations of smart contracts on traditional blockchains with respect to speed, storage costs, and computational capacity. This allows smart contracts for the rst time to implement fully decentralized applications that are hosted end to end on blockchain. The IC consists of a set of cryptographic protocols that connects independently operated nodes into a collection of blockchains. These blockchains host and execute \canisters", the IC's form of smart contracts. Canisters can store data, perform very general computations on that data, and provide a complete technology stack, serving web pages directly to end users. Computational and storage costs are covered by a \reverse-gas model", where canister developers pre-pay costs in cycles that are obtained from ICP, the native token of the IC. ICP tokens are also used for governance: the IC is governed by a decentralized autonomous organization, or DAO, which, among other things, determines changes to the topology of the network and upgrades to the protocol.

智能合同是一种新形式的软件，它将彻底改变软件的编写、 IT 系统的维护以及应用程序和整个企业的构建方式。智能合同是一种可组合的、自主的软件，它运行在去中心化的区块链上，这使得它们具有防篡改和不可阻挡的特性。本文介绍了一种全新的区块链设计——互联网计算机，它充分发挥了智能契约的潜力，克服了传统区块链在速度、存储成本和计算能力等方面的局限性。这允许智能合同第一次实现完全分散的应用程序，托管端到端的区块链。IC 由一组密码协议组成，它们将独立操作的节点连接到一组区块链中。这些区块链主机和执行罐“ ，IC 的智能合同的形式。Canisters 可以存储数据，对数据进行非常一般的计算，并提供一个完整的技术堆栈，直接为终端用户提供网页服务。计算和储存成本由反向气体模型涵盖”，其中罐开发商预付费用的周期是从 ICP (集成电路的本地标记)获得的。ICP 令牌也用于治理: IC 由一个分散的自治组织(DAO)治理，DAO 决定了网络拓扑的更改和协议的升级。

1 Introduction
引言

1.1 Unleashing smart contracts
1.1 释放智能合同

Because of their unique features, smart contracts are the key enabler of Web3, the new approach to the web where applications are fully controlled by their users and run on decentralized blockchain platforms. Such decentralized applications (dapps) are typically tokenized, meaning tokens are distributed to users as rewards for participating in the dapps. Participation can come in many di erent forms, ranging from moderating and providing content to governing a dapp and to creating and maintaining a dapp. Usually, tokens can also be bought on exchanges; indeed, selling tokens is a common way to nance dapp development. Finally, tokens are also used as a form of payment for the services or contents a dapp o ers. Smart contracts running on today's blockchain platforms, including all the
由于其独特的功能，智能合同是 web3 的关键推动因素，web3 是一种新的 web 方法，应用程序完全由用户控制，并在分散的区块链平台上运行。这样的分散式应用程序(dapps)通常是标记化的，这意味着标记被分发给用户作为参与 dapps 的奖励。参与可以有很多不同的形式，从管理和提供内容到管理 dapp 和创建和维护 dapp。通常，令牌也可以在交易所买到; 事实上，出售令牌是一种常见的 dapp 开发方式。最后，令牌也被用作 dapp 用户的服务或内容的支付形式。Smart 契约运行在今天的区块链平台上，包括所有的

*https://dfinity.org/foundation/

* https://dfinity.org/foundation/

popular ones (such as Ethereum), su er from many limitations, such as high transaction and storage costs, slow computational speed, and the inability to serve frontends to users. As a result, many popular blockchain applications are not fully decentralized but are hybrids where most of the application is hosted on traditional cloud platforms and call out to smart contracts on a blockchain for a small part of their overall functionality. Unfortunately, this renders such applications non-decentralized, and opens them to many of the drawbacks of traditional cloud-hosted applications, such as being at the mercy of cloud providers, and being vulnerable to many single points of failure.
受到许多限制，如交易和存储成本高，计算速度慢，无法为用户提供前端服务。因此，许多流行的区块链应用程序并不是完全分散的，而是混合的，其中大部分应用程序托管在传统的云平台上，并在区块链上调用智能合同，以获得其总体功能的一小部分。不幸的是，这使得这些应用程序非分散化，并且使它们暴露出传统云托管应用程序的许多缺点，例如受云提供商的摆布，并且容易受到许多单点故障的影响。

The Internet Computer (IC) is a new platform for executing smart contracts. Here, we use the term \smart contract" in a very broad sense: a general-purpose, im-mutable, tamperproof computer program whose execution is performed autonomously on a decentralized public network.
互联网计算机(IC)是一个执行智能合同的新平台。在这里，我们使用术语“智能合同”在一个非常广泛的意义上：一个通用的，不可变的，防篡改的计算机程序，其执行是在一个分散的公共网络上自主执行。

- By general purpose, we mean that the class of smart contract programs is Turing complete (i.e., anything computable can be computed by a smart contract).
  通常，我们的意思是智能契约程序类是 Turing 完全的(即，任何可计算的东西都可以通过智能契约来计算)。

- By immutable, we mean that, once deployed, the code of a smart contract cannot be changed by a party unilaterally.[1]
  通过不可变，我们的意思是，一旦部署，智能合同的代码不能被一方单方面改变

- By tamperproof, we mean that the instructions of the program are carried out faithfully and that intermediate and nal results are accurately stored and/or transmitted.
  通过防篡改，我们意味着程序的指令被忠实地执行，中间和最终结果被准确地存储和/或传输。

- By autonomous, we mean that a smart contract is executed automatically by the network, without the need for any action on the part of any individual.
  通过自治，我们意味着智能合同是由网络自动执行的，不需要任何个人的任何行动。

- By a decentralized public network, we mean of network of computers that is publicly accessible, geographically distributed, and not under the control of a small number of individuals or organizations.
  我们所说的分散式公共网络，是指可以公开访问的计算机网络，这种计算机网络在地理上分布，不受少数个人或组织的控制。

In addition, smart contracts
此外，智能合同

- are composable, meaning that they may interact with one another, and

是可组合的，这意味着它们可以相互作用，并且

- support tokenization, meaning that they may use and trade digital tokens.

支持标记化，这意味着他们可以使用和交易数字标记。

Compared to existing smart contract platforms, the IC is designed to:
与现有的智能合约平台相比，集成电路的设计目的是:

- be more cost e ective, in particular, allowing applications to compute and store data at a fraction of the cost of previous platforms;

更具成本效益，特别是允许应用程序以前平台成本的一小部分计算和存储数据;

- provide higher throughput and lower latency for processing smart contract transactions;

为处理智能合同事务提供更高的吞吐量和更低的延迟;

- be more scalable, in particular, the IC can process unbounded volumes of smart con-tract data and computation natively because it can grow in capacity by adding more nodes to the network.

具有更高的可扩展性，特别是集成电路可以本地处理无限量的智能合同数据和计算，因为它可以通过向网络增加更多的节点来增加容量。

[1]The IC allows a range of mutability policies for smart contracts, ranging from purely immutable to unilaterally upgradable, with other options in between.
集成电路允许智能合同的一系列可变性政策，从纯粹的不可变性到单方面的可升级，以及介于两者之间的其他选项。

In addition to providing a smart contract platform, the IC is designed to act as a complete technology stack, such that systems and services can be built that run entirely on the IC. In particular, smart contracts on the IC can service HTTP requests created by end users, so that smart contracts can directly serve interactive web experiences. This means that systems and services can be created without relying on corporate cloud hosting services or private servers, thus providing all of the bene ts of smart contracts in a true end-to-end fashion.

除了提供一个智能合同平台，集成电路被设计成一个完整的技术堆栈，这样系统和服务就可以完全在集成电路上运行。特别是，IC 上的智能合同可以为终端用户创建的 HTTP 请求提供服务，因此智能合同可以直接为交互式 web 体验提供服务。这意味着可以在不依赖公司云托管服务或私有服务器的情况下创建系统和服务，从而以真正的端到端方式提供智能合同的所有优点。

Realizing the vision of Web3. For end-users, accessing IC-based services is largely transparent. Their personal data is more secure than when accessing applications on a public or private cloud, but the experience of interacting with the application is the same.
实现 web3 的愿景。对于终端用户来说，访问基于 ic 的服务在很大程度上是透明的。他们的个人数据比在公共或私有云上访问应用程序更安全，但与应用程序交互的体验是一样的。

For the people creating and managing those IC-based services, however, the IC elimi-nates many of the costs, risks, and complexities associated with developing and deploying modern applications and microservices. For example, the IC platform provides an alterna-tive to the consolidation driven by large technology companies that are monopolizing the Internet. In addition, its secure protocol guarantees reliable message delivery, transparent accountability, and resilience without relying on rewalls, backup facilities, load balancing services, or failover orchestration.

然而，对于创建和管理这些基于 IC 的服务的人来说，IC 消除了与开发和部署现代应用程序和微服务相关的许多成本、风险和复杂性。例如，IC 平台为垄断互联网的大型科技公司的整合提供了另一种选择。此外，它的安全协议保证了可靠的消息传递、透明的问责制和弹性，而不依赖于重新隔离、备份设施、负载平衡服务或故障转移编排。

Building the IC is about restoring the Internet to its open, innovative, and creative roots
建设集成电路就是要恢复互联网的开放性、创新性和创造性

| in other words, to realize the vision of Web3. To focus on a few speci c examples, the IC does the following:
换句话说，实现 web3 的愿景。为了集中讨论一些特殊的例子，集成电路做了以下工作:

- Supports interoperability, shared functions, permanent APIs, and ownerless applica-tions, all of which reduce platform risk and encourages innovation and collaboration.
支持互操作性、共享功能、永久 api 和无所有者应用程序，所有这些都降低了平台风险，鼓励创新和协作。

- Persists data automatically in memory, which eliminates the need for database servers and storage management, improves computational e ciency, and simpli es software development.
将数据自动存储在内存中，从而消除了对数据库服务器和存储管理的需求，提高了计算效率，简化了软件开发。

- Simpli es the technology stack that IT organizations need to integrate and manage, which improves operational e ciency

Simpli 是 IT 组织需要集成和管理的技术栈，它提高了运营效率

1.2 High level view of the Internet Computer
1.2 互联网计算机的高层视图

To a rst approximation, the IC is a network of interacting replicated state machines. The notion of a replicated state machine is a fairly standard concept in distributed systems [Sch90], but we give a brief introduction here, beginning with the notion of a state machine.
按照第一种近似，IC 是一个由相互作用的复制状态机组成的网络。复制状态机的概念在分布式系统中是一个相当标准的概念[Sch90]，但是我们在这里给出了一个简单的介绍，从状态机的概念开始。

A state machine is a particular model of computation. Such a machine maintains a state, which corresponds to main memory or other forms of data storage in an ordinary computer. Such a machine executes in discrete rounds: in each round, it takes an input, applies a state transition function to the input and the current state, obtaining an output and a new state. The new state becomes the current state in the next round.

状态机是一种特殊的计算模型。这种机器维持一种状态，这种状态对应于普通计算机的主存储器或其他形式的数据存储。这样的机器以离散的回合执行: 在每个回合中，它接受一个输入，对输入和当前状态应用一个状态转换函数，获得一个输出和一个新的状态。新的状态在下一轮中成为当前状态。

The state transition function of the IC is a universal function, meaning that some of the inputs and data stored in the state may be arbitrary programs which act on other inputs and data. Thus, such a state machine represents a general (i.e., Turing complete) model of computation.

集成电路的状态转换函数是一个通用函数，这意味着存储在状态中的一些输入和数据可能是作用于其他输入和数据的任意程序。因此，这样的状态机代表了一个通用(即图灵完整)的计算模型。

To achieve fault tolerance, the state machine may be replicated. A replicated state machine comprises a subnet of replicas, each of which is running a copy of the same state machine. A subnet should continue to function | and to function correctly | even if some replicas are faulty.

为了实现容错，状态机可以被复制。一个复制的状态机包含一个复制的子网，每个子网都运行着同一个状态机的一个副本。一个子网应该继续运行 | 并且正常运行 | 即使一些副本是错误的。

It is essential that each replica in a subnet processes the same inputs in the same order. To achieve this, the replicas in a subnet must run a consensus protocol [Fis83], which ensures that all replicas in a subnet process inputs in the same order. Therefore, the internal state of each replica will evolve over time in exactly the same way, and each replica will produce exactly the same sequence of outputs. Note that an input to a replicated state machine on the IC may be an input generated by an external user, or an output generated by another replicated state machine. Similarly, an output of a replicated state machine may be either an output directed to an external user, or an input to another replicated state machine.

子网中的每个副本必须以相同的顺序处理相同的输入。为了实现这一点，子网中的副本必须运行一个共识协议[Fis83] ，该协议确保子网进程中的所有副本以相同的顺序输入。因此，每个副本的内部状态将以完全相同的方式随着时间的推移而发展，每个副本将产生完全相同的输出序列。注意，IC 上复制状态机的输入可以是外部用户生成的输入，也可以是另一个复制状态机生成的输出。同样，一个复制状态机的输出可以是一个直接指向外部用户的输出，也可以是另一个复制状态机的输入。

1.3 Fault Models
1.3 故障模型

In this area of computer science, one typically considers two types of replica failures: crash faults and Byzantine faults. A crash fault occurs when a replica abruptly stops and does not resume. Byzantine faults are failures in which a replica may deviate in an arbitrary way from its prescribed protocol. Moreover, with Byzantine faults, one or possibly several replicas may be directly under the control of a malicious adversary who may coordinate the behavior of these replicas. Of the two types of faults, Byzantine faults are potentially far more disruptive.

在计算机科学领域，人们通常考虑两种类型的副本故障: 崩溃故障和拜占庭故障。当一个副本突然停止并且不再继续时，就会发生崩溃错误。拜占庭故障是指副本可能以任意方式偏离其规定协议的故障。此外，对于拜占庭错误，一个或者几个副本可能直接处于恶意对手的控制之下，恶意对手可能协调这些副本的行为。在这两种类型的故障中，拜占庭故障可能更具破坏性。

Protocols for consensus and for realizing replicated state machines typically make as-sumptions about how many replicas may be faulty and to to what degree (crash or Byzantine) they may be faulty. In the IC, the assumption is that if a given subnet has n replicas, then less than n=3 of these replicas are faulty and these faults may be Byzantine. (Note that the di erent subnets in the IC may have di erent sizes.)

协商一致和实现复制状态机的协议通常假设有多少个复制可能出错，以及它们可能出错的程度(崩溃或拜占庭)。在集成电路中，假设一个给定的子网有 n 个副本，那么这些副本中的错误少于 n = 3，这些错误可能是拜占庭式的。(请注意，IC 中不同的子网可能有不同的大小

1.4 Communication Models
1.4 通讯模式

Protocols for consensus and for implementing replicated state machines also typically make assumptions about the communication model, which characterizes the ability of an ad-versary to delay the delivery of messages between replicas. At opposite ends of the spectrum, we have the following models:
用于协商一致和实现复制状态机的协议通常也对通信模型做出假设，该模型描述了对手在复制之间延迟消息传递的能力。在频谱的两端，我们有以下模型:

- In the synchronous model, there exists some known nite time bound , such that for any message sent, it will be delivered in less than time .
  在同步模型中，存在一些已知的夜间时间限制，因此对于发送的任何消息，它将在不到一个时间内交付。

- In the asynchronous model, for any message sent, the adversary can delay its delivery by any nite amount of time, so that there is no bound on the time to deliver a message; however, each message must eventually be delivered.
  在异步模型中，对于发送的任何消息，对手可以将其传递延迟任何夜间时间，因此在传递消息的时间上没有界限; 但是，最终必须传递每个消息。

Since the replicas in an IC-subnet are typically distributed around the globe, the synchronous communication model would be highly unrealistic. Indeed, an attacker could
由于 IC-subnet 中的副本通常分布在全球各地，因此同步通信模型将非常不切实际。事实上，攻击者可以

compromise the correct behavior of the protocol by delaying honest replicas or the commu-nication between them. Such an attack is generally easier to mount than gaining control over and corrupting an honest replica.
通过延迟诚实的副本或者它们之间的通信来破坏协议的正确行为。这样的攻击通常比控制和破坏一个诚实的副本更容易实施。

In the setting of a globally distributed subnet, the most realistic and robust model is the asynchronous model. Unfortunately, there are no known consensus protocols in this model that are truly practical (more recent asynchronous consensus protocols, as in [MXC$^+$16], attain reasonable throughput, but not very good latency). So like most other practical Byzantine fault tolerant systems that do not rely on synchronous communication (e.g., PBFT [CL99, BKM18, AMN$^+$20]), the IC opts for a compromise: a partial synchrony communication model [DLS88]. Such partial synchrony models can be formulated in various ways. The partial synchrony assumption used by the IC says, roughly speaking, that for each subnet, communication among replicas in that subnet is periodically synchronous for short intervals of time; moreover, the synchrony bound does not need to be known in advance. This partial synchrony assumption is only needed to ensure that the consensus protocol makes progress (the so-called liveness property). The partial synchrony assumption (nor, for that matter, the eventual delivery assumption) is not needed to ensure correct behavior of consensus (the so-called safety property), nor is it needed anywhere else in the IC protocol stack.

在全球分布的子网环境中，最真实、最健壮的模型是异步模型。不幸的是，在这个模型中没有真正实用的已知共识协议(最近的异步共识协议，如[MXC + 16] ，获得了合理的吞吐量，但是延迟不是很好)。因此，像大多数其他实用的拜占庭容错系统，不依赖于同步通信(例如，PBFT [CL99，BKM18，AMN + 20]) ，集成电路选择了折衷: 部分同步通信模型[DLS88]。这种部分同步模型可以以各种方式制定。IC 所使用的部分同步假设大致说明，对于每个子网，该子网中副本之间的通信在短时间内是周期性同步的; 此外，同步界不需要事先知道。这种部分同步假设只是为了确保协商一致协议取得进展(即所谓的活性特性)。不需要部分同步假设(也不需要最终交付假设)来确保协商一致的正确行为(所谓的安全属性) ，也不需要它在 IC 协议栈的其他任何地方。

Under the assumption of partial synchrony and Byzantine faults, it is known that our bound of f < n=3 on the number of faults is optimal.
在部分同步和拜占庭故障的假设下，我们知道我们的 f < n = 3 的故障数量的界限是最优的。

1.5 Permission Models
1.5 许可模式

The earliest protocols for consensus (e.g., PBFT [CL99]) were permissioned, in the sense that the replicas comprising a replicated state machine are governed by a centralized or-ganization, which determines which entities provide replicas, the topology of the network, and possibly also implements some kind of centralized public-key infrastructure. Permis-sioned consensus protocols are typically the most e cient, and while they do avoid a single point of failure, the centralized governance is undesirable for certain applications, and it is antithetical to the spirit of the burgeoning Web3 era.
最早达成共识的协议(例如 PBFT [CL99])是被允许的，因为组成复制状态机的副本由一个集中的或组织来管理，这个集中的或组织决定哪些实体提供副本、网络的拓扑，并且可能还实现某种集中的公开金钥基础建设。Permis 协商一致协议通常是最古老的，虽然它们

确实避免了单一的故障点，但是对于某些应用程序来说，集中治理是不可取的，而且它与迅速发展的 web3 时代的精神背道而驰。

More recently, we have seen the rise of permissionless consensus protocols, such as Bitcoin [Nak08], Ethereum [But13], and Algorand [GHM+17]. Such protocols such are based on a blockchain and either a proof of work (PoW) (e.g., Bitcoin, Ethereum prior to v2.0) or a proof of stake (PoS) (e.g., Algorand, Ethereum v2.0) . While such protocols are completely decentralized, they are much less e cient than permissioned protocols. We also point out that, as observed in [PSS17], PoW-based consensus protocols such as Bitcoin cannot guarantee correctness (i.e., safety) in an asynchronous communication network.

最近，我们看到了无许可共识协议的兴起，比如 Bitcoin [Nak08]、 Ethereum [But13]和 algor [GHM + 17]。这样的协议基于一个区块链和一个工作证明(PoW)(例如 Bitcoin，Ethereum before v2.0)或一个赌注证明(PoS)(例如 Algorand，Ethereum v2.0)。虽然这些协议是完全去中心化的，但是它们远远不如被允许的协议成熟。我们还指出，正如在[PSS17]中所观察到的，像 Bitcoin 这样的基于 pow- 的共识协议不能保证异步通信网络中的正确性(即安全性)。

The IC’s permission model is a hybrid model, obtaining the e ciency of a permissioned protocol while o ering many of the bene ts of a decentralized PoS protocol. This hybrid model is called a DAO-controlled network and (roughly speaking) works as follows: each subnet runs a permissioned consensus protocol, but a decentralized autonomous organization (DAO) determines which entities provide replicas, con gures the topology of the network, provides a public-key infrastructure, and controls which version of the protocol is deployed to the replicas. The IC’s DAO is called the network nervous system (NNS), and is based on a PoS, so that all decisions taken by the NNS are made by community

集成电路的权限模型是一个混合模型，它在获得权限协议的效率的同时，也获得了分散的 PoS 协议的许多优点。这种混合模型被称为 DAO 控制的网络(大致来说) ，其工作原理如下:每个子网运行一个被允许的共识协议，但是一个分散的自治组织(DAO)确定哪些实体提供副本，控制网络的拓扑结构，提供一个公开金钥基础建设，并控制哪个版本的协议被部署到副本。集成电路的 DAO 被称为网络神经系统(NNS) ，它基于一个 PoS，因此由 NNS 做出的所有决策都是由社区做出的

members whose voting power is determined by how much of the IC's native governance token they have staked in the NNS (see Section 1.8 for more in this token). Through this PoS-based governance system, new subnets can be created, replicas may be added to or removed from existing subnets, software updates may be deployed, and other modi cations to the IC may be e ected. The NNS is itself a replicated state machine, which (like any other) runs on a particular subnet whose membership is determined via the same PoS-based governance system. The NNS maintains a database called the registry, which keeps track of the topology of the IC: which replicas belong to which subnets, the public keys of the replicas, and so on. (See Section 1.10 for a few more details on the NNS.)
其投票权取决于他们在 NNS 中投入了多少 IC 的本机治理令牌(参见第 1.8 节了解更多关于此令牌的内容)。通过这个基于 pos- 的治理系统，可以创建新的子网，可以在现有子网中添加或删除副本，可以部署软件更新，可以对 IC 进行其他修改。NNS 本身是一个复制的状态机，它(像其他任何机器一样)运行在一个特定的子网上，子网的成员资格是通过相同的基于 pos- 的治理系统确定的。NNS 维护一个称为注册中心的数据库，它跟踪 IC 的拓扑结构: 哪些副本属于哪个子网，副本的公钥，等等。(参见第 1.10 节了解更多关于 NNS 的细节

Thus, one sees that the IC's DAO-controlled network allows the IC to achieve many of the practical bene ts of a permissioned network (in terms of more e cient consensus), while maintaining many of the bene ts of a decentralized network (with governance controlled by a DAO).

因此，人们可以看到，IC 的 DAO 控制网络允许 IC 实现许多被许可的网络的实际优点(在更为科学的共识方面)，同时保持许多分散式网络的优点(治理由 DAO 控制)。

The replicas running the IC protocol are hosted on servers in geographically distributed, independently operated data centers. This also bolsters the security and decentralized nature of the IC.

运行 IC 协议的副本托管在地理上分布的、独立操作的数据中心的服务器上。这也加强了 IC 的安全性和分散性。

1.6 Chain-key cryptography
1.6 链式密钥加密

The IC's consensus protocol does, in fact, use a blockchain, but it also uses public-key cryptography, speci cally, digital signatures: the registry maintained by the NNS is used to bind public keys to replicas and subnets as a whole. This enables a unique and powerful collection of technologies that we call chain-key cryptography, which has several components.
事实上，IC 的共识协议确实使用区块链，但它也使用公开密钥加密，特别是数字签名: NNS 维护的注册表用于将公开密钥与副本和子网整体绑定。这使得一个独特而强大的技术集合，我们称之为链式密钥加密，它有几个组成部分。

1.6.1 Threshold signatures
1.6.1 阈值签名

The rst component of chain-key cryptography is threshold signatures: this is a well established cryptographic technique that allows a subnet to have a public signature-veri cation key whose corresponding secret signing key is split into shares, which are distributed among all of the replicas in a subnet in such a way that the shares held by the corrupt replicas do not let them forge any signatures, while the shares held by the honest replicas allow the subnet to generate signatures consistent with the policies and protocols of the IC.

链式密钥加密的第一个组成部分是门限签名：这是一种公认的加密技术，它允许一个子网拥有一个公共签名验证密钥，其相应的秘密签名密钥被分成若干份，这些份分布在子网中的所有副本之间，这样，腐败副本所持有的份不会让它们伪造任何签名，而诚实副本所持有的份则允许子网生成与 IC 的策略和协议一致的签名。

One critical application of these threshold signatures is that
这些门限签名的一个关键应用是

> an individual output of one subnet may be veri ed by another subnet or external user by simply validating a digital signature with respect to the public signature-veri cation key of the (rst) subnet.
> 一个子网的单个输出可以由另一个子网或外部用户通过简单地验证与(第一个)子网的公共签名验证密钥有关的数字签名来验证。

Note that the public signature-veri cation key for a subnet may be obtained from the NNS
请注意，子网的公共签名验证密钥可以从 NNS 获得

| this public signature-veri cation key remains constant over the lifetime of a subnet (even as the membership of a subnet may change over that lifetime). This stands in contrast to many non-scalable blockchain-based protocols, which require the entire blockchain to be validated in order to validate any single output.
这个公共签名验证密钥在子网的生命周期内保持不变(即使子网的成员资格在该生命周期内可能发生变化)。这与许多不可扩展的基于区块链的协议形成对比，这些协议需要验证整个区块链以验证任何单个输出。

As we will see, these threshold signatures have a number of other applications within the IC. One such application is to give each replica in a subnet access to unpredictable
正如我们将要看到的，这些阈值签名在 IC 中有许多其他的应用。其中一个应用就是给子网中的每个副本提供不可预测的访问

pseudorandom bits (derived from such signatures). This is the basis for the Random Beacon used in consensus and the Random Tape used in execution.
伪随机位(从这种签名派生而来)。这就是在共识中使用的 Random Beacon 和在执行中使用的 Random Tape 的基础。

In order to securely deploy threshold signatures, the IC uses an innovative distributed key generation (DKG) protocol that constructs a public signature-veri cation key and provisions each replica with a share of the corresponding secret signing key, and works within our fault and communication model.
为了实现门限签名的安全部署，集成电路采用了一种创新的分布式密钥生成(DKG)协议，该协议构造了一个公共签名验证密钥，并为每个副本提供了相应的秘密签名密钥的一部分，该协议在我们的故障和通信模型中工作。

1.6.2 Chain-evolution technology
1.6.2 链式进化技术

Chain-key cryptography also includes a sophisticated collection of technologies for robustly and securely maintaining a blockchain based replicated state machine over time, which together form what we call chain-evolution technology. Each subnet operates in epochs of many rounds (typically on the order of a few hundreds of rounds). Using threshold signatures and a number of other techniques, chain-evolution technology implements many essential maintenance activities that are executed periodically with a cadence that is tied to epochs:
链密钥加密技术还包括一系列复杂的技术，用于随着时间的推移稳健和安全地维护基于区块链的复制状态机，这些技术共同构成了我们所说的链进化技术。每个子网以多轮的时代运行(通常在几百轮的顺序上)。使用阈值签名和许多其他技术，链式进化技术实现了许多基本的维护活动，这些维护活动以与时代相关的节奏周期性地执行:

Garbage collection: At the end of each epoch, all inputs that have been processed, and all consensus-level protocol messages needed to order those inputs, may safely be purged from the memory of each replica. This is essential in keeping the storage requirements for the replicas from growing without bound. This is in contrast to many non-scalable blockchain-based protocols, where the entire blockchain from the genesis block must be stored.
垃圾收集: 在每个时代结束时，所有已经处理的输入以及所有需要对这些输入进行排序的共识级协议消息都可以安全地从每个副本的内存中清除。这对于保证副本的存储需求不会无限制地增长至关重要。这与许多不可扩展的基于区块链的协议形成了鲜明的对比，在这些协议中，来自起源区块的整个区块链必须被存储。

Fast forwarding: If a replica in a subnet falls very far behind its peers (because it is down or disconnected from the network for a long time), or a new replica is added to a subnet, it can be fast forwarded to the beginning of the most recent epoch, without having to run the consensus protocol and process all of the inputs up to that point. This is in contrast to many non-scalable blockchain-based protocols, where the entire blockchain from the genesis block must be processed.
快速转发: 如果子网中的一个副本远远落后于它的同伴(因为它长时间处于停机状态或与网络断开连接)，或者一个新的副本被添加到一个子网中，那么它就可以快速转发到最近一个纪元的开始，而不必运行共识协议并处理到那个时候为止的所有输入。这与

许多不可扩展的基于区块链的协议形成了鲜明对比，在这些协议中，来自起源区块的整个区块链都必须进行处理。

Subnet membership changes: The membership of the subnet (as determined by the NNS, see Section 1.5) may change over time. This can only happen at an epoch boundary, and needs to be done with care to ensure consistent and correct behavior.

子网成员资格的变化: 子网的成员资格(由 NNS 决定，参见 1.5 节)可能随着时间的推移而变化。这只能发生在一个时代的边界上，并且需要小心的做，以确保一致和正确的行为。

Pro-active resharing of secrets: We mentioned above in Section 1.6.1 how the IC uses chain-key cryptography | speci cally, threshold signatures | for output veri cation. This is based on secret sharing, which avoids any single point of failure by splitting up a secret (in this case, a secret signing key) into shares that are stored among the replicas. At the beginning of each epoch, these shares are pro-actively reshared. This achieves two goals:

主动重新共享秘密: 我们在上面的 1.6.1 节中提到了 IC 如何使用链式密钥加密技术，特别是门限签名来进行输出验证。这是基于秘密共享的，它通过将秘密(在本例中为秘密签名密钥)分解为存储在副本之间的共享来避免任何单点故障。在每个新纪元的开始，这些共享被主动地重新分享。这实现了两个目标:

- When the membership of a subnet changes, the resharing will ensure that any new members have an appropriate share of the secret, while any replicas that are no longer members no longer have a share of the secret.

  当子网的成员更改时，重新共享将确保任何新成员拥有适当的秘密份额，而任何不再是成员的副本不再拥有秘密份额。

- If a small number of shares are leaked to an attacker in any one epoch, or even in every epoch, those shares will not do an attacker any good.

  如果在任何一个时代，甚至每个时代都有少量的共享泄露给攻击者，那么这些共享对攻击者没有任何好处。

Protocol upgrades: When the IC protocol itself needs to be upgraded, to x bugs or add new features, this can be done automatically using a special protocol at the beginning of an epoch.

协议升级: 当 IC 协议本身需要升级到 x bug 或添加新特性时，这可以在新纪元开始时使用特殊协议自动完成。

1.7 Execution Models

1.7 执行模型

As already mentioned, replicated state machines in the IC can execute arbitrary programs. The basic computational unit in the IC is called a canister, which is roughly the same as the notion of a process, in that it comprises both a program and its state (which changes over time).

如前所述，IC 中的复制状态机可以执行任意程序。集成电路中的基本计算单元称为“容器”，它与进程的概念大致相同，因为它包括程序及其状态(随着时间的推移而变化)。

Canister programs are encoded in WebAssembly, or Wasm for short, which is a binary instruction format for a stack-based virtual machine. Wasm is an open standard.[2] While it was initially designed to enable high-performance applications on web pages, it is actually very well suited to general-purpose computation.

Canister 程序被编码在 WebAssembly 中，简称 Wasm，是一种基于堆栈的虚拟机的二进制指令格式。Wasm 是一个开放标准。2 虽然它最初是为了在网页上实现高性能应用程序而设计的，但实际上它非常适合通用计算。

The IC provides a run-time environment for executing Wasm programs in a canister, and to communicate with other canisters and external users (via message passing). While, in principle, one can write a canister program in any language that may be compiled to Wasm, a language called Motoko has been designed that is well aligned with the operational semantics of the IC. Motoko is a strongly typed, actor-based[3] programming language with built-in support for orthogonal persistence[4] and asynchronous message passing. Orthogo-nal persistence simply means that all memory maintained by a canister is automatically persisted (i.e., it does not have to be written to a le). Motoko has a number of productiv-ity and safety features, including automatic memory management, generics, type inference, pattern matching, and both arbitrary and xed-precision arithmetic.

IC 提供了一个运行时环境，用于在一个容器中执行 Wasm 程序，并与其他容器和外部用户通信(通过消息传递)。虽然在原则上，人们可以用任何语言编写一个可以编译到 Wasm 的罐式程序，但是一种名为 Motoko 的语言已经被设计出来，它与 IC 的操作语义非常一致。Motoko 是一种强类型的，基于 actor 的 3 编程语言，内置对正交持久性和异步消息传递的支持。Orthogo-nal 持久性仅仅意味着由一个容器维护的所有内存被自动持久化(即，它不必写入一个 le)。Motoko 具有自动内存管理、泛型、类型推理、模式匹配以及任意和定精度算法等生产性和安全性特点。

In addition to Motoko, the IC also provides a messaging interface de nition language and wire format called Candid, for typed, high-level, and cross-language interoperability. This allows any two canisters, even if written in di erent high-level languages, to easily communicate with one another.

除了 Motoko 之外，IC 还提供了一种称为 Candid 的消息接口定义语言和有线格式，用于类型化、高级别和跨语言的互操作性。这允许任何两个罐子，即使写在不同的高级语言，很容易相互沟通。

To fully support canister development in any given programming language, besides a Wasm compiler for that language, certain run-time support must also be provided. At the

present time, in addition to Motoko, the IC also fully supports canister development in the Rust programming language.

为了完全支持任何给定编程语言中的罐式开发，除了该语言的 Wasm 编译器之外，还必须提供某些运行时支持。目前，除了 Motoko 之外，IC 也完全支持 Rust 编程语言的容器开发。

1.8 Utility token
1.8 工具令牌

The IC makes use of a utility token called ICP. This token is used for the following functions:

集成电路使用一个名为 ICP 的实用令牌。这个令牌用于以下功能:

Staking in the NNS: As already discussed in Section 1.5, ICP tokens may be staked in the NNS to acquire voting rights so as to participate in the DAO that controls the IC network. Users that have ICP tokens staked in the NNS and who participate in

在 NNS 中: 正如在第 1.5 节中已经讨论过的，ICP 令牌可以被放在 NNS 中以获得投票权，从而参与控制 IC 网络的 DAO。拥有 ICP 令牌并参与 NNS 的用户

2See https://webassembly.org/.

参见 https://webassembly.org/。

3See https://en.wikipedia.org/wiki/Actor_model.

参见 https://en.wikipedia.org/wiki/Actor _ model。

4See https://en.wikipedia.org/wiki/Persistence_(computer_science)#Orthogonal_or_transparent_persistence.

4 参见 https://en.wikipedia. org/wiki/persistence _ (computer _ science) # 正交或透明的持久性。

the NNS governance also receive newly minted ICP tokens as a voting reward. The amount of the award is determined by policies established and enforced by the NNS.
NNS 治理也会收到新铸造的 ICP 令牌作为投票奖励。奖励的金额由 NNS 制定和执行的政策决定。

Conversion to Cycles: ICP is used to pay for the usage of the IC. More speci cally, ICP tokens can be converted to cycles (i.e., burned), and these cycles are used to pay for creating canisters (see Section 1.7) and for the resources that canisters use (storage, CPU, and bandwidth). The rate at which ICP is converted to cycles is determined by the NNS.
转换为循环: ICP 用于支付 IC 的使用费。更具体地说，ICP 令牌可以转换为循环(即燃烧)，这些循环用于支付创建容器(参见 1.7 节)和容器使用的资源(存储、 CPU 和带宽)。ICP 转换为循环的速率由 NNS 决定。

Payment to Node Providers: ICP tokens are used to pay the node providers|these are the entities that own and operate the computing nodes that host the replicas that make up the IC. At regular intervals (currently monthly), the NNS decides on the number of newly minted tokens that each node provider should receive, and sends the tokens to the node provider's account. Payment of tokens is conditioned on providing reliable service to the IC, according to speci c policies established and enforced by the NNS.
向节点提供商付款: ICP 令牌用于向节点提供商付款 | 这些实体拥有并操作承载组成 IC 的副本的计算节点。NNS 定期(当前是每月一次)决定每个节点提供者应该接收的新生成的令牌的数量，并将令牌发送到节点提供者的帐户。令牌的支付取决于向 IC 提供可靠的服务，根据 NNS 建立和强制执行的特定政策。

1.9 Boundary Nodes
1.9 边界节点

Boundary nodes provide the network edge services of the IC. In particular, they o er
边界节点提供集成电路的网络边缘服务

- clearly de ned entry points to the IC,
  很明显是进入 IC 的入口，
- denial of service protection for the IC,
  分布式拒绝服务攻击，
- seamless access to the IC from legacy clients (e.g., web browsers).
  从遗留客户端(例如网页浏览器)无缝接达 IC。

To facilitate seamless access to the IC from a legacy client, boundary nodes provide func-tionality to translate a standard HTTPS request from a user to an ingress message directed toward a canister on the IC, and then route this ingress message to speci c replicas on the subnet where this canister resides. Furthermore, boundary nodes o er additional services to improve the user experience: caching, load balancing, rate limiting, and the ability for legacy clients to authenticate responses from the IC.
为了方便遗留客户机对 IC 的无缝访问，边界节点提供功能，将用户的标准 HTTPS 请求转换为指向 IC 上的一个容器的入口消息，然后将这个入口消息路由到容器所在的子网上

的特定复制品。此外，边界节点或其他服务可以改善用户体验: 缓存、负载平衡、速率限制，以及遗留客户端验证 IC 响应的能力。

A canister is identi ed by a URL on the ic0.app domain. Initially, a legacy client looks up the corresponding DNS record for the URL, obtains an IP address of a boundary node, and then sends an initial HTTPS request to this address. The boundary node returns a javascript-based \service worker" that will be executed in the legacy client. After this, all interactions between the legacy client and the boundary node will be done via this service worker.

罐子是通过 ic0.app 域上的 URL 来识别的。最初，遗留客户机查找相应的 DNS 记录以获取 URL，获取边界节点的 IP 地址，然后向该地址发送初始 HTTPS 请求。边界节点返回一个基于 javascript 的服务工作者”，这将在遗留客户端中执行。在这之后，遗留客户端和边界节点之间的所有交互都将通过这个服务工作者完成。

One of the essential tasks carried out by the service worker is to authenticate responses from the IC using chain-key cryptography (see Section 1.6). To do this, the public veri ca-tion key for the NNS is hard-coded in the service worker.

服务工作者执行的一项基本任务是使用链式密钥加密法对 IC 的响应进行身份验证(参见第 1.6 节)。为了做到这一点，NNS 的公共验证密钥在服务工作者中是硬编码的。

The boundary node itself is responsible for routing requests to a replica on the subnet on which the speci ed canister is hosted. The information needed to perform this routing is obtained by the boundary node from the NNS. The boundary node keeps a list with replicas that provide timely replies and selects a random replica from this list.

边界节点本身负责将请求路由到托管指定容器的子网上的副本。执行此路由所需的信息由边界节点从 NNS 获得。边界节点保持一个包含副本的列表，这些副本可以提供及时的回复，并从这个列表中选择一个随机的副本。

All communication between legacy clients and boundary nodes and between boundary nodes and replicas is secured via TLS.[5]

遗留客户端和边界节点之间以及边界节点和副本之间的所有通信都是通过 TLS 保护的

In addition to legacy clients, it is also possible to interact with boundary nodes using \IC native" clients, which already include the service-worker logic, and do not need to retrieve the service worker program from the boundary node.

除了遗留客户端之外，还可以使用 IC 本机”客户端与边界节点交互，这些客户端已经包含服务工作者逻辑，并且不需要从边界节点检索服务工作者程序。

Just as for replicas, the deployment and con guration of boundary nodes is controlled by the NNS.

就像副本一样，边界节点的部署和配置是由 NNS 控制的。

1.10 More details of the NNS
1.10 更多关于 NNS 的资料

As already mentioned in Section 1.5, the network nervous system (NNS) is an algorithmic governance system that controls the IC. It is realized by a set of canisters on a special system subnet. This subnet is like any other subnet, but is con gured somewhat di erently (as one example, canisters on the system subnet are not charged for the cycles they use).

正如在 1.5 节中已经提到的，网络神经系统(NNS)是一个控制 IC 的算法治理系统。它是通过在一个特殊的系统子网上的一组罐子来实现的。这个子网和其他子网一样，但是它的设计有些不同(例如，系统子网上的罐子使用的周期是不收费的)。

Some of the most relevant NNS canisters are

一些最相关的 NNS 罐是

- the registry canister, which stores the con guration of the IC, i.e., which repli-cas belong to which subnet, the public keys associated with subnets and individual replicas, and so on.

  存储 IC 的配置的注册表容器，即哪个副本属于哪个子网、与子网关联的公钥和单个副本，等等。

- the governance canister, which manages the decision making and voting on how the IC should be evolved, and

  管理机构，负责管理关于 IC 应如何发展的决策和投票

- the ledger canister, which keeps track of the users’ ICP utility token accounts and the transactions between them.

  分类账容器，用于跟踪用户的 ICP 实用代币账户和他们之间的交易。

1.10.1 Decision making on the NNS
1.10.1 有关 NNS 的决策

Anyone can participate in NNS governance by staking ICP tokens in so-called neurons. Neuron holders can then suggest and vote on proposals, which are suggestions on how the IC should be changed, e.g., how the subnet topology or the protocol should be changed. The neurons’ voting power for decision making is based on proof of stake. Intuitively, neurons with more staked ICP tokens have more voting power. However, the voting power also depends on some other neuron characteristics, e.g., more voting power is given to neuron holders that are committed to keep their tokens staked for a longer period of time.

任何人都可以通过在所谓的神经元中放置 ICP 标记来参与 NNS 的治理。然后，神经元持有者可以建议并投票表决提案，这些提案是关于如何更改 IC 的建议，例如，如何更改子网拓扑或协议。神经元对决策的投票权是基于利害关系的证明。直观地说，拥有更多利害关系的 ICP 令牌的神经元有更多的投票权。然而，投票权也取决于一些其他神经元特征，例如，更多的投票权给予神经元持有者，承诺保持他们的令牌更长的时间。

Each proposal has a determined voting period. A proposal is adopted if at the voting period's end, a simple majority of the total voting power has voted in favor of the proposal and these Yes-votes constitute a given quorum (currently 3%) of the total voting power. Otherwise, the proposal is rejected. In addition, a proposal is adopted or rejected at any point if an absolute majority (more than half of the total voting power) is in favor or against the proposal, respectively.

每个提案都有一个确定的投票期。如果在投票期结束时，总投票权的简单多数投票赞成该提案，且这些赞成票构成总投票权的特定法定人数(目前为 3%)，则通过该提案。否则，提案将被否决。此外，如果绝对多数(占总表决权的一半以上)分别赞成或反对某项提案，则该提案在任何时候均获得通过或否决。

If a proposal is adopted, the governance canister automatically executes the decision. For example, if a proposal suggests changing the network topology and is adopted, the governance canister automatically updates the registry with the new con gurations.

如果一个提案被采纳，管理机构将自动执行该决定。例如，如果一个建议建议更改网络拓扑并被采纳，那么治理罐将自动更新注册表，并使用新的密码。

5See https://en.wikipedia.org/wiki/Transport_Layer_Security.
参见 https://en.wikipedia. org/wiki/transport _ layer _ security。

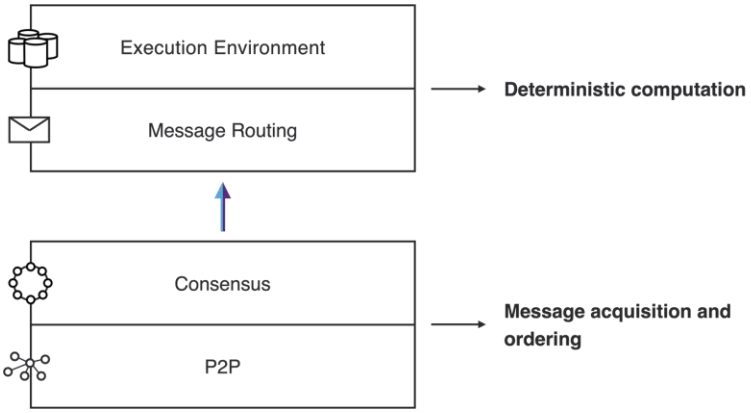

Figure 1: The layers of the Internet Computer Protocol
图 1: Internet 计算机协议的层

1.11 Work in progress
1.11 正在进行的工作

The architecture of the IC is still evolving and expanding. Here are a few new features that will be deployed soon:
集成电路的体系结构仍在发展和扩展。以下是一些即将部署的新功能:

DAO-controlled canisters. Just like the overall con guration of the IC is controlled by the NNS, any canister may also be controlled by its own DAO, called the service nervous system (SNS). The DAO controlling a canister can control updates to the canister logic, as well as issuing privileged commands to be carried out by the canister.

Dao 控制的罐子。就像集成电路的整体结构由神经网络控制一样，任何集成电路也可能由其自身的 DAO (称为服务神经系统(SNS))控制。控制一个滤毒罐的 DAO 可以控制滤毒罐逻辑的更新，以及发出由滤毒罐执行的特权命令。

Threshold ECDSA. ECDSA signatures [JMV01] are used in cryptocurrencies, such as Bitcoin and Ethereum, as well as in many other applications. While threshold signa-tures are already an essential ingredient in the IC, these are not threshold ECDSA signatures. This new feature will allow individual canisters to control an ECDSA sign-ing key, which is securely distributed among all of the replicas on the subnet hosting the canister.

阈值 ECDSA。ECDSA 签名[JMV01]用于加密货币，如比特币和以太网，以及许多其他应用程序。虽然阈值签名已经是 IC 中的一个重要组成部分，但这些不是阈值 ECDSA 签名。这个新功能将允许个人罐控制 ECDSA 签名密钥，这是安全地分布在所有副本的子网托管的罐子。

Bitcoin and Ethereum integration. Building on the new threshold ECDSA feature, this feature will allow canisters to interact with the Bitcoin and Ethereum blockchains, including the ability to sign transactions.

比特币和以太网的整合。基于新的 ECDSA 阈值功能，该功能将允许金属罐与比特币和以太区块链交互，包括签署交易的能力。

HTTP integration. This feature will allow canisters to read arbitrary web pages (external to the IC).
HTTP 集成。这个功能将允许罐读取任意网页(外部的集成电路)。

2 Architecture overview
架构概述

As illustrated in Figure 1, the Internet Computer Protocol consists of four layers:
如图 1 所示，因特网计算机协议包括四个层次:

- peer-to-peer layer (see Section 4);
  对等层(见第 4 节) ;
- consensus layer (see Section 5);
  共识层(见第 5 节) ;
- routing layer (see Section 6);
  路由层(见第 6 节) ;

- execution layer (see Section 7).
  执行层(见第 7 节)。

Chain-key cryptography impacts several layers, and is discussed in detail in Sections 3 (threshold signatures) and 8 (chain-evolution technology).
链密钥密码学影响几个层次，并在第 3 节(门限签名)和第 8 节(链进化技术)中详细讨论。

2.1 Peer-to-peer layer
2.1 对等层

The peer-to-peer layer's task is to transport protocol messages between the replicas in a subnet. These protocol messages consist of
对等层的任务是在子网中的副本之间传输协议消息。这些协议消息包括

- messages used to implement consensus,
  用于实现共识的消息，

- input messages generated by an external user.
  输入由外部用户生成的消息。

Basically, the service provided by the peer-to-peer is a \best e ort" broadcast channel:
基本上，点对点提供的服务是一个最好的广播频道:

> if an honest replica broadcasts a message, then that message will eventually be received by all honest replicas in the subnet.
> 如果一个诚实的副本广播一条消息，那么该消息最终将被子网中所有诚实的副本接收。

Design goals include the following:
设计目标包括:

- Bounded resources. All algorithms must work with bounded resources (memory, bandwidth, CPU).
  有限资源。所有算法都必须使用有限资源(内存、带宽、 CPU)。

- Prioritization. Di erent messages may be treated with di erent priorities, depend-ing on certain attributes (e.g., type, size, round), and these priorities may change over time.
  优先顺序。根据某些属性(例如，类型、大小、圆形) ，不同的消息可以使用不同的优先级进行处理，并且这些优先级可能随着时间的推移而改变。

- E ciency. High throughput is more important than low latency.
  高吞吐量比低延迟更重要。

- DOS/SPAM resilience. Corrupt replicas should not prevent honest replicas from communicating with one another.
  DOS/SPAM 弹性。腐败的副本不应该阻止诚实的副本之间的通信。

2.2 Consensus layer
2.2 共识层

The job of the consensus layer of the IC is to order inputs so that all replicas in a subnet will process such inputs in the same order. There are many protocols in the literature for this problem. The IC uses a new consensus protocol, which is described here at a high level. Any secure consensus protocol should guarantee two properties, which (roughly stated)
IC 的共识层的工作是对输入进行排序，这样子网中的所有副本将以相同的顺序处理这些输入。在这个问题的文献中有许多协议。集成电路使用一个新的共识协议，这里在高层描述。任何安全的协商一致协议都应该保证两个属性，这两个属性(粗略地说)
are:
是:

- safety: all replicas in fact agree on the same ordering of inputs, and
安全性: 所有副本实际上都同意相同的输入顺序，并且

- liveness: all replicas should make steady progress.
活性: 所有副本都应该取得稳定的进展。

The IC consensus protocol is designed to be
IC 共识协议的设计目的是

- extremely simple, and
非常简单，而且

- robust: performance degrades gracefully when some replicas are malicious.
: 当一些副本是恶意的时候，性能会优雅地下降。

As discussed above, we assume f < n=3 faulty (i.e., Byzantine) replicas. Also, liveness holds under a partial synchrony assumption, while safety is guaranteed, even in a completely asynchronous network.

如上所述，我们假设 f < n = 3 个错误的(即拜占庭)复制品。此外，活性在部分同步假设下保持，同时安全性得到保证，即使在完全异步的网络中也是如此。

Like a number of consensus protocols, the IC consensus protocol is based on a blockchain. As the protocol progresses, a tree of blocks is grown, starting from a special genesis block that is the root of the tree. Each non-genesis block in the tree contains (among other things) a payload, consisting of a sequence of inputs, and a hash of the block's parent in the tree. The honest replicas have a consistent view of this tree: while each replica may have a di erent, partial view of this tree, all the replicas have a view of the same tree. In addition, as the protocol progresses, there is always a path of nalized blocks in this tree. Again, the honest replicas have a consistent view of this path: while each replica may have a di erent, partial view of this path, all the replicas have a view of the same path. The inputs in the payloads of the blocks along this path are the ordered inputs will be processed by the execution layer of the Internet Computer.

与许多协商一致协议一样，IC 协商一致协议也是基于区块链的。随着协议的进展，一个块树开始生长，从一个特殊的起源块开始，这个起源块是树的根。树中的每个非起源块包含(除其他外)一个有效负载，由输入序列和树中块的父块的散列组成。诚实的副本对该树具有一致的视图: 尽管每个副本可能具有该树的不同部分视图，但所有副本都具有相同树的视图。此外，随着协议的进展，这棵树中总会有一条 nalized 块的路径。同样，诚实的副本具有这条路径的一致视图: 尽管每个副本可能具有这条路径的不同部分视图，但所有副本都具有相同路径的视图。沿着这条路径的块的有效负载中的输入是由 Internet 计算机的执行层处理的有序输入。

The protocol proceeds in rounds. In round h of the protocol, one or more blocks of height h are added to the tree. That is, the blocks added in round h are always at a distance of exactly h from the root. In each round, a pseudo-random process is used to assign each replica a unique rank, which is an integer in the range 0; : : : ; n 1. This pseudo-random process is implemented using a Random Beacon (this makes use of threshold signatures, mentioned above in Section 1.6.1 and discussed in more detail in Section 3). The replica of lowest rank is the leader of that round. When the leader is honest and the network is synchronous, the leader will propose a block, which will be added to the tree; moreover, this will be the only block added to the tree in this round and it will extend the nalized path. If the leader is not honest or the network is not synchronous, some other replicas of higher rank may also propose blocks, and also have their blocks added to the tree. In any case, the logic of the protocol gives highest priority to the leader's proposed block and some block or blocks will be added to this tree in this round. Even if the protocol proceeds for a few rounds without extending the nalized path, the height of the tree will continue to grow with each round, so that when the nalized path is extended in round h, the nalized path will be of length h. A consequence of this, even if the latency occasionally increases because of faulty replicas or unexpectedly high network latency, the throughput of the protocol remains essentially constant.

协议循环进行。在协议的第一轮 h 中，一个或多个高度 h 块被添加到树中。也就是说，在第一轮 h 中添加的块始终与根 h 保持精确的距离。在每个回合中，使用伪随机过程为每个副本分配一个唯一的秩，即范围为 0: : : ; n1 的整数。这个伪随机过程是使用 Random Beacon 实现的(这利用了阈值签名，在上面的 1.6.1 节中提到，在第 3 节中进行了更详细的讨论)。最低排名的副本是这一轮的领导者。当领导者诚实且网络是同步的时候，领导者会提出一个块，

这个块将被添加到树中，而且这将是这一轮中唯一添加到树中的块，它将扩展 nalized 路径。如果领导者不诚实或网络不同步，其他一些级别较高的副本也可能提出块，并将它们的块添加到树中。在任何情况下，该协议的逻辑给予最高优先级的领导者提出的块，一些块或块将添加到该树在这一轮。即使协议进行了几个回合而没有扩展 nalized 路径，树的高度也会随着每个回合继续增长，因此当 nalized 路径在第 h 回合扩展时，nalized 路径的长度将为 h。这样的结果是，即使由于错误的副本或意外的高延迟，延迟时间偶尔会增加，协议的吞吐量基本上保持不变。

The consensus protocol relies on digital signatures to authenticate messages sent between replicas. To implement the protocol, each replica is associated with a public veri cation key for a signature scheme. The association of replicas to public keys is obtained from the registry maintained by the NNS.

协商一致协议依赖于数字签名来验证副本之间发送的消息。为了实现该协议，每个副本与一个签名方案的公共验证密钥相关联。副本与公钥的关联从 NNS 维护的注册中心获得。

2.3 Message routing
2.3 消息路由

As discussed in Section 1.7, basic computational unit in the IC is called a canister. The IC provides a run-time environment for executing programs in a canister, and to communicate with other canisters and external users (via message passing).
如第 1.7 节所讨论的，集成电路中的基本计算单元称为容器。集成电路提供了一个运行时环境，用于在一个容器中执行程序，并与其他容器和外部用户通信(通过消息传递)。

The consensus layer bundles inputs into payloads, which get placed into blocks, and as blocks are nalized, the corresponding payloads are delivered to the message routing layer, then processed by the execution environment, which updates the state of the

共识层将输入捆绑到有效负载中，这些有效负载被放置到块中，并且随着块的固定化，相应的有效负载被交付到消息路由层，然后由执行环境进行处理，执行环境将更新

canisters on the replicated state machine and generates outputs, and these outputs are processed by the message routing layer.
在复制的状态机上的罐子并生成输出，这些输出由消息路由层处理。

It is useful to distinguish between two types of inputs:
区分两种类型的输入是有用的:

ingress messages: these are messages from external users;
输入消息: 这些是来自外部用户的消息;

cross-subnet messages: these are messages from canisters on other subnets.
跨子网消息: 这些消息来自其他子网上的罐子。

We can also distinguish between two types of outputs:
我们还可以区分两种类型的输出:

ingress message responses: these are responses to ingress messages (which may be re-trieved by external users);
入口消息响应: 这些是对入口消息的响应(外部用户可以重新检索这些消息) ;

cross-subnet messages: these are messages to canisters on other subnets.
跨子网消息: 这些是其他子网上的邮件。

Upon receiving a payload from consensus, the inputs in that payload are placed into various input queues. For each canister C running on a subnet, there are several in-put queues: one for ingress messages to C, and for each other canister C^0 with whom C communicates, one for cross-subnet messages to C from C^0.
从协商一致接收有效负载后，该有效负载中的输入被放置到各种输入队列中。对于在子网上运行的每个罐 c，有几个输入队列: 一个用于向 c 输入消息，另一个用于与 c 通信的其他罐 C0，一个用于跨子网消息从 c0 到 c。

In a each round, the execution layer will consume some of the inputs in these queues, update the replicated state of the relevant canisters, and place outputs in various output queues. For each canister C running on a subnet, there are several output queues: for each other canister C^0 with whom C communicates, one for cross-subnet messages to C^0 from C. The message routing layer will take the messages in these output queues and place them into subnet-to-subnet streams to be processed by an crossnet transfer protocol, whose job it is to actually transport these messages to other subnets.
在每一轮中，执行层将使用这些队列中的一些输入，更新相关罐的复制状态，并将输出放在各种输出队列中。对于在子网上运行的每个罐 c，有几个输出队列: 对于 c 与之通信的每个罐 C0，一个用于跨子网消息从 c 发送到 C0。消息路由层将获取这些输出队列中的消息，并将它们放入子网到子网的流中，由交叉网传输协议处理，交叉网传输协议的工作实际上是将这些消息传输到其他子网。

In addition to these output queues, there is also an ingress history data structure. Once an ingress message has been processed by a canister, a response to that ingress message will be recorded in this data structure. At that point, the external user who provided the ingress message will be able to retrieve the corresponding response. (Note that ingress history does not maintain the full history of all ingress messages.)
除了这些输出队列，还有一个入口历史记录数据结构。一旦进入消息被一个小罐处理，对该进入消息的响应将被记录在这个数据结构中。在这一点上，提供进入消息的外部用户将能够检索相应的响应。(请注意，入口历史并不保存所有入口消息的完整历史

Note that the replicated state comprises the state of the canisters, as well as \system state", including the above-mentioned queues and streams, as well as the ingress history data structure. Thus, both the message routing and execution layers are involved in updating and maintaining the replicated state of a subnet. It is essential that all of this state is updated in a completely deterministic fashion, so that all replicas maintain exactly the same state.

请注意，复制的状态包括罐的状态以及系统状态”，包括上述队列和流以及入口历史数据结构。因此，消息路由和执行层都涉及到更新和维护子网的复制状态。所有这些状态都必须以一种完全确定的方式更新，这样所有的副本才能保持完全相同的状态。

Also note that the consensus layer is decoupled from the message routing and execution layers, in the sense that any forks in the consensus blockchain are resolved before their payloads are passed to message routing, and in fact, consensus does not have to keep in lock step with message routing and consensus and is allowed to run a bit ahead.

还请注意，共识层与消息路由和执行层是分离的，因为共识区块链中的任何分叉在其有效负载被传递给消息路由之前就已经解析了，而且实际上，共识不必与消息路由和共识保持同步，并且允许稍微提前一点。

2.3.1 Per-round certi ed state
每轮认证状态 2.3.1

In each round, some of the state of a subnet will be certi ed. The per-round certi ed state is certi ed using chain-key cryptography. Among other things, the certi ed state in a given round consists of

在每一轮中，子网的某些状态将被确认。每一轮的认证状态都使用链式密钥加密技术进行认证。其中，给定一轮中的认证状态包括

- cross-subnet messages that were recently added to the subnet-to-subnet streams;
最近添加到子网到子网流中的跨子网消息;

- other metadata, including the ingress history data structure.
其他元数据，包括入口历史数据结构。

The per-round certi ed state is certi ed using a threshold signature (see Section 1.6.1).
每轮认证状态使用阈值签名进行认证(参见第 1.6.1 节)。
Per-round certi ed state is used in several ways in the IC:
每轮认证状态在集成电路中有几种用法:

- Output authentication. Cross-subnet messages and responses to ingress messages are authenticated using per-round certi ed state.
输出认证。交叉子网消息和对入口消息的响应使用每轮认证状态进行认证。

- Preventing and detecting non-determinism. Consensus guarantees that each replica processes inputs in the same order. Since each replica processes these inputs determin-istically, each replica should obtain the same state. However, the IC is designed with an extra layer of robustness to prevent and detect any (accidental) non-deterministic computation, should it arise. The per-round certi ed state is one of the mechanisms used to do this.
防止和检测非确定性。Consensus 保证每个副本以相同的顺序处理输入。由于每个副本确定性地处理这些输入，每个副本应该获得相同的状态。然而，IC 设计了一个额外的健壮性层，以防止和检测任何(偶然的)非确定性计算，如果它出现。每轮确认状态是用来做到这一点的机制之一。

- Coordination with consensus. The per-round certi ed state is also used to coordinate the execution and consensus layers, in two di erent ways:
协调一致。每轮认证状态也用于协调执行和共识层面，有两种不同的方式:

  { If consensus is running ahead of execution (whose progress is determined by the last round whose state is certi ed), consensus will be \throttled".
  如果共识在执行之前运行(其进度取决于最后一轮的状态验证)，共识将被扼杀。”。

  { Inputs to consensus must pass certain validity checks, and these validity checks may depend on certi ed state, which all replicas must agree upon.
  {共识的输入必须通过某些有效性检查，这些有效性检查可能依赖于已验证的状态，所有副本都必须对此达成一致。

2.3.2 Query calls vs update calls
2.3.2 查询调用 vs 更新调用

As we have described it so far, an ingress messages must pass through consensus so that they are processed in the same order by all replicas on a subnet. However, an important optimization is available to those ingress messages whose processing does not modify the replicated state of a subnet. These are called query calls | as opposed to other ingress messages, which are called update calls. Query calls are allowed to perform computations which read and possibly update the state of a canister, but any updates to the state of a canister are never committed to the replicated state. As such, a query call may be processed directly by a single replica without passing through consensus, which greatly reduces the latency for obtaining a response from a query call.

正如我们到目前为止所描述的，入口消息必须通过共识，以便子网上的所有副本以相同的顺序处理它们。然而，一个重要的优化是可用于那些进入消息的处理没有修改子网的复制状态。这些被称为查询调用 | 与其他入口消息不同，后者被称为更新调用。允许查询调用执行读取并可能更新罐状态的计算，但是对罐状态的任何更新都不会提交给复制状态。因此，查询调用可以由单个副本直接处理，而无需通过共识，这大大减少了从查询调用获得响应的延迟。

In general, a response to a query call is not recorded in the ingress history data structure, and therefore cannot be authenticated using the per-round certi ed state mechanism as described above. However, the IC makes it possible for canisters to store data (while processing update calls) in special certi ed variables, which can be authenticated by this mechanism; as such, query calls that return as their value a certi ed variable can still be authenticated.

通常，对查询调用的响应不记录在入口历史数据结构中，因此不能使用上述每轮验证状态机制进行身份验证。然而，集成电路使罐子能够将数据(在处理更新调用时)存储在特殊的经过验证的变量中，这些变量可以通过这种机制进行验证；因此，返回经过验证的变量作为其值的查询调用仍然可以进行验证。

2.3.3 External user authentication
2.3.3 外部用户身份验证

One of the main di erences between an ingress message and a cross-subnet message is the mechanism used for authenticating these messages. While chain-key cryptography is used
入口消息和跨子网消息的主要区别之一是用于认证这些消息的机制。当使用链式密钥加密时

to authenticate cross-subnet messages, a di erent mechanism is used to authenticate ingress messages from external users.
为了验证跨子网消息，一个不同的机制被用来验证从外部用户进入的消息。

There is no central registry for external users. Rather, an external user identi es himself to a canister using a user identi er, which is a hash of a public signature-veri cation key. The user holds a corresponding secret signing key, which is used to sign ingress messages. Such a signature, as well as the corresponding public key, is sent along with the ingress message. The IC automatically authenticates the signature and passes the user identi er to the appropriate canister. The canister may then authorize the requested operation, based on the user identi er and other parameters to the operation speci ed in the ingress message.

没有针对外部用户的中心注册表。相反，一个外部用户使用一个用户标识符(一个公共签名验证密钥的散列)将自己标识到一个容器中。用户持有相应的秘密签名密钥，用于签署进入消息。这样的签名，以及相应的公钥，与进入消息一起发送。IC 会自动认证签名，并将用户标识传递给相应的容器。然后，基于用户标识符和进入信息中指定的操作的其他参数，容器可以授权请求的操作。

First-time users generate a key pair and derive their user identi er from the public key during their rst interaction with the IC. Returning users are authenticated using the secret key that is stored by the user agent. A user may associate several key pairs with a single user identity, using signature delegation. This is useful, as it allows a single user to access the IC from several devices using the same user identity.

第一次用户生成一个密钥对，并在第一次与 IC 交互时从公钥中获得用户标识。返回用户使用用户代理存储的密钥进行身份验证。一个用户可以使用签名委托将几个密钥对与一个用户身份关联起来。这很有用，因为它允许一个用户使用相同的用户标识从几个设备访问 IC。

2.4 Execution layer
2.4 执行层

The execution layer processes one input at a time. This input is taken from one of the input queues, and is directed to one canister. Based on this input and the state of the canister, the execution environment updates the state of the canister, and additionally may add messages to output queues and update the ingress history (possibly with a response to an earlier ingress message).
执行层一次处理一个输入。这个输入是从其中一个输入队列中获取的，然后被引导到一个罐子上。执行环境根据这个输入和容器的状态更新容器的状态，另外还可以向输出队列添加消息并更新进入历史记录(可能是对早期进入消息的响应)。

Each subnet has access to a distributed pseudorandom generator (PRG). Pseudo-random bits are derived from a seed that itself is a threshold signature called the Random Tape (see Section 1.6.1 and more detail in Section 3). There is a di erent Random Tape for each round of the consensus protocol.
每个子网都可以访问一个分布式伪随机生成器(PRG)。伪随机位是从一个种子派生出来的，种子本身就是一个叫做 Random Tape 的阈值签名(参见第 1.6.1 节和第 3 节中的更多细节)。每一轮协商一致的协议都有一个不同的 Random Tape。

The basic properties of the random tape are:
随机磁带的基本属性是:

1. Before a block at height h is nalized by any honest replica, the Random Tape at height h + 1 is guaranteed to be unpredictable.
在任何诚实副本对高度 h 的块进行 nalized 之前，高度 h + 1 的 Random Tape 保证是不可预测的。

2. By the time block at height h + 1 is nalized by any honest replica, that replica will typically have all the shares it needs to construct the Random Tape at height h + 1.
通过高度 h + 1 的时间块被任何诚实的副本纳化，该副本通常拥有在高度 h + 1 构造 Random Tape 所需的所有共享。

To obtain pseudorandom bits, a subnet must make a request for these bits via a \system call" from the execution layer in some round, say h. The system will then respond to that request later, using the Random Tape at height h+1. By property (1) above, it is guaranteed that the requested pseudorandom bits are unpredictable at the time the request is made. By property (2) above, the requested random bits will typically be available at the time the next block is nalized.

为了获得伪随机位，子网必须通过一个系统调用从执行层发出对这些位的请求。系统稍后会响应这个请求，使用高度为 h + 1 的 Random Tape。通过上面的属性(1)，可以保证请求的伪随机位在请求发出时是不可预测的。上面的 By 属性(2)，请求的随机位通常在下一个块被 nalized 的时候是可用的。

2.5 Putting it all together
2.5 把它们放在一起

We trace through the typical ow to process a user request on the IC.
我们通过典型的 ow 来处理用户在 IC 上的请求。

1. A user's request M to a canister C is sent by the user's client to a boundary node (see Section 1.9), and the boundary node sends M to a replica on the subnet that hosts canister C.
用户对一个容器 c 的请求 m 被用户的客户端发送到一个边界节点(参见 1.9 节)，而边界节点将 m 发送到容器 c 所在的子网上的副本。

2. After receiving M, this replica will broadcast M to all other replicas on the subnet, using the peer-to-peer layer (see Section 2.1).
在接收到 m 之后，这个副本将使用对等层将 m 广播到子网上的所有其他副本(参见第 2.1 节)。

3. Having received M, the leader for the next round of consensus (see Section 2.2) will bundle M with other inputs to form the payload for a block B that the leader proposes.
收到 m 后，下一轮共识的领导者(见 2.2 节)将把 m 与其他输入捆绑在一起，形成领导者提出的 b 区块的有效载荷。

4. Some time later, block B is nalized and the payload is sent to the message routing layer (see Section 2.3) for processing. Note that the peer-to-peer layer is also used by consensus to nalize this block.
一段时间后，块 b 被固定，有效负载被发送到消息路由层进行处理(见第 2.3 节)。请注意，点对点层也被一致使用来实现这个块。

5. The message routing layer will place M in the input queue of the canister C.
消息路由层将把 m 放在容器 c 的输入队列中。

6. Some time later, the execution layer (see Section 2.4) will process M, updating the internal state of the canister C.
一段时间后，执行层(参见 2.4 节)将处理 m，更新滤筒 c 的内部状态。

   In some situations, the canister C will be able to immediately compute a response R to the request M. In this case, R is placed in the ingress history data structure.
   在某些情况下，罐 c 将能够立即计算对请求 m 的响应 r。在这种情况下，r 被放置在入口历史数据结构中。

   In other situations, processing the request M may require making a request to a
   在其他情况下，处理 m 请求可能需要向

   another canister. In this example, let us suppose that to process this particular request M, the canister C must make a request M^0 to another canister C^0 that resides on another subnet. This second request M^0 will be placed in the output queue of the C, and then the following steps are performed.
   另一个毒气罐。在这个例子中，让我们假设要处理这个特定请求 m，罐 c 必须向驻留在另一个子网上的另一个罐 c0 发出请求 M0。第二个请求 m0 将被放置在 c 的输出队列中，然后执行以下步骤。

7. Some time later, message routing will move M^0 into an appropriate cross-subnet stream, and this will eventually be transported to the subnet hosting C^0.
一段时间后，消息路由会将 m0 移动到一个适当的跨子网流中，并最终传输到子网托管 C0。

8. On the second subnet, the request M^0 will be obtained from the rst subnet, and eventually pass through consensus and message routing on the second subnet and
在第二个子网上，请求 m0 将从第一个子网获得，并最终通过第二个子网上的一致性和消息路由

   then be processed by execution. The execution layer will update the internal state of canister C^0 and generate a response R^0 to the request M^0. The response R^0 will go in the output queue of canister C^0, and eventually be placed in a cross-subnet stream and transported back to the rst subnet.

然后执行处理。执行层将更新罐子 c0 的内部状态，并对请求 m0 生成响应 R0。响应 r0 将进入容器 c0 的输出队列，并最终被放置在一个跨子网流中，并传输回第一个子网。

9. Back on the rst subnet, the response R^0 will be obtained from the second subnet, and eventually pass through consensus and message routing on the rst subnet and then be processed by execution. The execution layer will update the internal state of canister C and generate a response R to the original request message M. This response will be recorded in the ingress history data structure.

在第一个子网上，将从第二个子网获得响应 R0，并最终通过第一个子网上的共识和消息路由，然后执行处理。执行层将更新罐 c 的内部状态，并对原始请求消息 m 生成响应 r。这个响应将记录在入口历史数据结构中。

Regardless of which execution path is taken, the response R to request M will eventually be recorded in the ingress history data structure on the subnet that hosts canister C. To obtain this response, the user's client must perform a kind of \query call" (see Section 2.3.2). As discussed in Section 2.3.1, this response will be authenticated via chain-key cryptography (speci cally, using a threshold signature). The authentication logic itself (i.e., threshold signature veri cation) will be performed by the client using the service worker originally obtained by the client from the boundary node.

无论采用哪种执行路径，请求 m 的响应 r 最终将记录在托管罐 c 的子网上的入口历史数据结构中。为了获得这个响应，用户的客户端必须执行一种查询调用”(参见第 2.3.2 节)。正如第 2.3.1 节所讨论的，这个响应将通过链式密钥加密(特别是使用阈值签名)进行身份验证。身份验证逻辑本身(即阈值签名验证)将由客户端使用最初由客户端从边界节点获得的服务工作者执行。

3 Chain-key crytography I: threshold signatures
链式密钥加密 i: 阈值签名

A critical component of the IC's chain-key cryptography is a threshold signature scheme [Des87]. The IC uses threshold signatures for a number of purposes. Let n be the number of replicas in a subnet and let f be a bound on the number of corrupt replicas.
集成电路的链式密钥加密的一个关键组成部分是门限签名方案[Des87]。IC 将阈值签名用于许多目的。N 表示子网中副本的数量，f 表示损坏副本的数量。

- The Consensus Layer makes use of an (f + 1)-out-of-n threshold signature to realize a random beacon (see Section 5.5).
  共识层利用(f + 1)-out-of-n 阈值签名来实现随机信标(参见第 5.5 节)。

- The Execution Layer makes use of an (f + 1)-out-of-n threshold signature to realize a random tape, which is used to provide unpredictable pseudorandom numbers to canisters (see Section 7.1).
  执行层利用一个(f + 1)-out-of-n 阈值签名来实现一个随机磁带，该磁带用于向罐子提供不可预测的伪随机数生成器(见第 7.1 节)。

- The Execution Layer makes use of an (n f)-out-of-n threshold signature to certify the replicated state. This is used both to authenticate the outputs of a subnet (see Section 6.1) and to implement the fast-forwarding feature of the IC's chain-evolution technology (see Section 8.2).
  执行层使用(n f)-out-of-n 阈值签名来验证复制的状态。这既用于验证子网的输出(参见第 6.1 节)，也用于实现 IC 的链式进化技术的快进特性(参见第 8.2 节)。

For the rst two applications (the random beacon and random tape), it is essential that the threshold signatures are unique, i.e., for a given public key and message, there is only one valid signature. This is required as we use the signature as a seed to a pseudorandom generator, and all replicas who compute such a threshold signature must agree on the same seed.
对于前两个应用程序(随机信标和随机磁带)，阈值签名必须是唯一的，即对于给定的公钥和消息，只有一个有效的签名。这是必需的，因为我们使用签名作为伪随机生成器的种子，并且所有计算这种阈值签名的副本必须在相同的种子上达成一致。

3.1 Threshold BLS signatures
3.1 阈值 BLS 签名

We implement threshold signatures based on the BLS signature scheme [BLS01], which is trivial to adapt to the threshold setting.
我们基于 BLS 签名方案[BLS01]实现了门限签名，这对于门限设置的适应是微不足道的。

The ordinary (i.e., non-threshold) BLS signature scheme makes use of two groups, G and G^0, both of prime order q. We assume that G is generated by g 2 G and G^0 is generated by g^0 2 G^0. We also assume a hash function H_{G^0} that maps its inputs to G^0 (and which is modeled as a random oracle). The secret signing key is an element x 2 Z_q and the public veri cation key is V := g^x 2 G.

普通(即非阈值) BLS 签名方案使用两组，g 和 G0，都是素数阶 q。我们假设 g 是由 g 2 g 生成的，g0 是由 g02 g0 生成的。我们还假设一个哈希函数 hg0 将其输入映射到 G0(并被建模为随机预言)。秘密签名键是一个元素 x 2 Zq，公共验证键是 v: = gx 2 g。

In the non-threshold setting, to sign a message m, the signer computes h^0 H_{G_0} (m) 2 G^0 and then computes the signature := $(h^0)^x$ 2 G^0. To verify that such a signature is valid, one must test if $\log_{h_0}$ = $\log_g$ V . To be able to perform this test e ciently, the BLS scheme uses the notion of a pairing on the groups G and G^0, which is a special algebraic tool that is available when G and G^0 are elliptic curves of a special type. We shall not be able to go into the details of pairings and elliptic curves here. See [BLS01] for more details. BLS signatures have the nice property (mentioned above) that signatures are unique.

在非阈值设置中，为了对消息 m 进行签名，签名者计算 h0 HG0(m)2 G0，然后计算签名: = (h0) x2 G0。为了验证这样的签名是否有效，必须测试 logh0 = logg v。为了更好地完成这个检验，BLS 方案在群 g 和 g0 上引入了配对的概念，这是当 g 和 g0 是特殊类型的椭圆曲线时可用的一种特殊的代数工具。我们不能在这里讨论配对和椭圆曲线的细节。更多细节见[BLS01]。BLS 签名有一个很好的属性(上面提到过)，即签名是唯一的。

In the t-out-of-n threshold setting, we have n replicas, any t of which may be used to generate a signature on a message. On somewhat more detail, each replica P_j holds a share x_j 2 Z_q of the secret signing key x 2 Z_q, which is privately held by P_j, while the group element V_j := g^{x_j} is publicly available. The shares (x_1; : : : ; x_n) are a t-out-of-n secret-sharing of x (see Section 3.4).

在 t-out-of-n 阈值设置中，我们有 n 个副本，其中任何 t 都可用于在消息上生成签名。更详细地说，每个副本 Pj 持有秘密签名密钥 x2 Zq 的一份 xj 2 Zq，这是 Pj 私有的，而组元素 Vj: = gxj 是公开可用的。股票(x1; : : ; xn)是 x 的 t-out-of-n 秘密共享(参见 3.4 节)。

Given a message m, replica P_j can generate a signature share

$$_j := (h^0)^{x_j} 2 G^0;$$

给定一条消息 m，副本 Pj 可以生成一个签名共享 j: = (h0)

xj2g0;

where $h^0 := H_{G_0}(m)$ as before. To verify that such a signature share is valid, one must test if $\log_{h_0} {}_j = \log_g V_j$. This can be done using a pairing, as discussed above | in fact, this is exactly the same as the validity test for an ordinary BLS signature with public key V_j.
其中 h0: = HG0(m)。为了验证这样的签名共享是否有效，必须测试 logh0 j = logg Vj。这可以使用配对来完成，如上所述 | 事实上，这与使用公钥 Vj 的普通 BLS 签名的有效性测试完全相同。

This scheme satis es the following reconstruction property:
该方案满足以下重构性质:

> Given any collection of t valid signature shares $_j$ on a message m (contributed by distinct replicas), we can e ciently compute a valid BLS signature on m under the public veri cation key.
> 给定消息 m (由不同副本提供)上的任何有效签名共享 j 的集合，我们可以在公共验证密钥下计算 m 上的有效 BLS 签名。
>
> In fact, can be computed as
> 事实上，可以计算为
>
> $$Y \quad {}_j{}^{j}; \qquad (1)$$
> $$Y\ j\ j\ (1)$$
> $$j$$
>
> where the $_j$'s can be e ciently computed just from the indices of the t contributing replicas.
> 其中 j 可以仅从 t 贡献副本的指数计算出来。

Under reasonable intractability assumptions for G, and modeling H_{G_0} as a random oracle, this scheme satis es the following security property:
在 g 的合理难解性假设下，将 hg0 建模为随机预言，该方案满足以下安全性质:

> Assume that at most f replicas may be corrupted by an adversary. Then it is infeasible for the adversary to compute a valid signature on a message unless it obtains signature shares on that message from at least t f honest replicas.
> 假设最多 f 个副本可能被对手破坏。那么对手在消息上计算一个有效的签名是不可行的，除非它从至少 t 个诚实的副本中获得该消息的签名份额。

3.2 Distributed key distribution
3.2 分布式密钥分配

To implement threshold BLS, we need a way to distribute the shares of the secret signing key to the replicas. One way to do this would be to have a trusted party compute all of these shares directly and distribute them to all the replicas. Unfortunately, this would create a single point of failure. Instead, we use a distributed key generation (DKG) protocol, which allows the replicas to essentially carry out the logic of such a trusted party using a secure distributed protocol.
为了实现阈值 BLS，我们需要一种将秘密签名密钥的份额分配给副本的方法。一种方法是让一个可信任的一方直接计算所有这些共享，然后将它们分发给所有的副本。不幸的是，这会造成一个单点故障。相反，我们使用分布式密钥生成(DKG)协议，该协议允许副本使用安全的分布式协议实现这种可信任方的逻辑。

We sketch the high level ideas of the protocol currently implemented. We refer the reader to [Gro21] for more details. The DKG protocol used is essentially non-interactive. It uses two essential ingredients:
我们概述了目前实现的协议的高层次思想。我们将读者参考[Gro21]了解更多细节。使用的 DKG 协议基本上是非交互式的。它使用了两个基本要素:

- a publicly veri able secret sharing (PVSS) scheme, and
一个公开可信的秘密共享(PVSS)计划，以及

- a consensus protocol.
共识协议。

Although any consensus protocol could be used, not surprisingly, the one we use is that in Section 5 (see also Section 8).
尽管任何共识协议都可以使用，但毫不奇怪，我们使用的是第 5 节中的协议(另见第 8 节)。

3.3 Assumptions
3.3 假设

The basic assumptions made are the same as outlined in Section 1:
作出的基本假设与第一节中概述的相同:

- asynchronous communication, and
异步通信，和

- f < n=3.
< n = 3.

We only indirectly make use of a partial synchrony assumption (as in Section 5.1) to ensure that the consensus protocol attains liveness.
我们只是间接地使用部分同步假设(如第 5.1 节)来确保协商一致协议获得活性。

We also assume that for a t-out-of-n threshold signature scheme, we have
我们还假设对于 t-out-of-n 门限签名方案，我们有

f < t n f;

F < n f;

which (among other things) ensures that (1) the corrupt replicas cannot sign all by themselves, and (2) the honest replicas can sign all by themselves.
确保(1)损坏的副本不能自己签名，(2)诚实的副本可以自己签名。

We also assume that every replica is associated with some public keys, where each replica also holds the corresponding private key. One public key is the signing key (the same one as in Section 5.4). Another public key is a public encryption key for a speci c public-key encryption scheme needed to implement the PVSS scheme (details follow).
我们还假设每个副本都与一些公钥相关联，其中每个副本也包含相应的私钥。一个公钥是签名密钥(与第 5.4 节中的相同)。另一个公钥是实现 PVSS 方案所需的特定公钥加密方案的公钥(详情如下)。

3.4 PVSS scheme
3.4 自订车辆登记号码计划

Let G be the group of prime order q generated by g 2 G introduced above. Let s 2 Z_q be secret.
Recall that a t-out-of-n Shamir secret-sharing of s is a vector (s1; : : : ; sn) 2 Znq, where
设 g 是由上面介绍的 g 2 g 生成的素数阶 q 的群。S 2 Zq 是秘密的。回想一下，s 的 t-out-of-n Shamir 秘密共享是向量(s1; : : : sn)2 Znq，其中

s_j := a(j) (j = 1; : : : ; n):
Sj: = a (j)(j = 1; : : ; n) :

and
和

a(x) := $a_0 + a_1x + + a_{t\ 1}x^{t\ 1}$ 2 Z_q[x]
A (x) : = a0 + a1x + + 在 1xt12zq [x]处

is a polynomial of degree less than t with a_0 := s. The key properties of such a secret sharing are
是小于 t 的多项式，其中 a0: = s。这种秘密共享的关键特性是

- from any collection of t of the s_j's, we can e ciently compute (via polynomial interpolation) the secret s = a_0 = a(0), and
从 sj 的任何 t 的集合中，我们可以很容易地计算(通过多项式插值)秘密 s = a0 = a (0)，并且

- if a_1; : : : ; $a_{t\ 1}$ are chosen uniformly and independently over Z_q, then any collection of fewer than t of the s_j's reveals no information about the secret s.
如果 a1; : : ; at 1 在 Zq 上被一致和独立地选择，那么任何小于 t 的 sj 的集合都不会显示关于秘密 s 的信息。

At a high level, a PVSS scheme allows one replica, P_i, called the dealer, to take such a sharing, and compute an object called a dealing, which contains

在高层次上，PVSS 方案允许一个称为交易商的副本 Pi 接受这样的共享，并计算一个称为交易的对象，该对象包含

- a vector of group elements $(A_0; : : : ; A_{t\ 1})$, where $A_k := g^{a_k}$ for k = 0; : : : ; t 1,
群元素的向量(A0; : : : ; At 1)，其中 k = 0 的 Ak: = gak;
- a vector of ciphertexts $(c_1; : : : ; c_n)$, where c_j is the encryption of s_j under P_j's public encryption key,
密文的向量(c1; : : : ; cn)，其中 cj 是 Pj 的公开密钥下的 sj 的加密,
- a non-interactive zero-knowledge proof that each c_j does indeed encrypt such a share | more precisely, that each c_j decrypts the value s_j satisfying
一个非交互的零知识证明，每个 cj 确实加密了这样一个共享 | 更精确，每个 cj 解密值 sj 满足

$$gsj = \prod_{k=0}^{t\ 1} Ajk = ga(j): \quad (2)$$

$$Gsj = \prod_{k=0}^{t\ 1} Ajk = ga\ (j) : \quad (2)$$

We note that to establish the overall security of our DKG protocol, the PVSS scheme must provide an appropriate level of chosen ciphertext security. Speci cally, the dealer must embed its identity as associated data in the dealing, and the encrypted shares must

我们注意到，为了建立 DKG 协议的整体安全性，PVSS 方案必须提供适当级别的选择密文安全性。特别地，经销商必须将其身份作为相关数据嵌入到交易中，并且加密的共享必须

remain hidden, even under a chosen ciphertext attack wherein an adversary is allowed to decrypt arbitrary dealings which are decrypted under associated data that is distinct from the associated data used to create the dealing.
即使在选定的密文攻击下，对手仍然可以解密任意交易，而这些交易是在与创建交易所使用的相关数据不同的相关数据下解密的。

It is easy to realize a PVSS scheme, if one is not too concerned about e ciency. The idea is to use an ElGamal-like encryption scheme to encrypt each s_j bit by bit, and then use a standard non-interactive zero-knowledge proof for the relation (2), which would be based on an a standard application of the Fiat-Shamir transform (see [FS86]) to an appropriate Sigma protocol (see [CDS94]). While this yields a polynomial-time scheme, it is not that practical. However, there are many possible ways to optimize this type of scheme. See [Gro21] for the details on the highly optimized PVSS scheme used in the IC.

如果一个人不太担心网络安全，那么 PVSS 方案就很容易实现。其思想是使用一种类似 elgamal 的加密方案来逐位加密每个 sj，然后对关系(2)使用一种标准的非交互式零知识证明，该证明将基于 Fiat-Shamir 变换(参见[FS86])对适当的 Sigma 协议的标准应用(参见[CDS94])。虽然这产生了一个多项式时间方案，但它并不那么实用。然而，有很多可能的方法来优化这种类型的方案。请参阅[Gro21]了解 IC 中使用的高度优化的 PVSS 方案的详细信息。

3.5 The basic DKG protocol
3.5 DKG 的基本协议

Using the PVSS scheme and a consensus protocol, the basic DKG protocol is very simple.
使用 PVSS 方案和共识协议，基本的 DKG 协议非常简单。

1. Each replica broadcasts a signed dealing of a random secret to all other replicas.
每个副本向所有其他副本广播一个随机秘密的签名交易。

   Such a signed dealing includes a dealing, along with the identity of the dealer and a signature on the dealing under the dealer's public signing key.
   这样的签名交易包括一个交易，以及交易商的身份和交易商的公共签名密钥下的签名。

   Such a signed dealing is called valid if it has the right syntactic form, and the signature and non-interactive zero knowledge proof are valid.
   这种签名交易如果具有正确的句法形式，且签名和非交互式零知识证明是有效的，则称之为有效交易。

2. Using consensus the replicas agree on a set S of f + 1 valid signed dealings (from distinct dealers).
使用协商一致，副本同意一组 f + 1 有效的签名交易(来自不同的交易商)。

3. Suppose that the ith dealing in the set S contains the vector of group elements $(A_{i;0}; : : : ; A_{i;t\ 1})$ and the vector of ciphertexts $(c_{i;1}; : : : ; c_{i;n})$.
假设集合 s 中的 th 处理包含群元素的向量(Ai; 0; : : ; Ai; t1)和密文的向量(ci; 1; : : ; ci; n)。

   Then the public veri cation key for the threshold signature scheme is
   那么门限签名方案的公开验证密钥是

$$V := \prod_i A_{i;0}:$$

Note that the secret signing key is implicitly de ned as
注意，秘密签名键被隐含地定义为

$$x := \log_g V:$$
$$: = \log g\ v:$$

P_j's share of the secret signing key x is

Pj 在秘密签名键 x 中的份额是

$$x_j := \sum_i s_{i;j};$$
$$Xj: = si;$$

where $s_{i;j}$ is the decryption of $c_{i;j}$ under P_j's secret decryption key.
其中 si; j 是 Pj 的秘密解密密钥下 ci; j 的解密。

The public veri cation key for replica P_j is
复制 Pj 的公共验证密钥为

$$V_j := \prod_{i=1}^{t} \prod_{k=0}^{T-1} A_{i;k}^{j} = g^{x_j} :$$
$$Vj: = y\ y\ Aji;\ kk = gxj:$$

Note that the shares x_j comprise a t-out-of-n Shamir secret-sharing of x. As such, the $_j$ values appearing in (1) are just Lagrange interpolation coe cients. This establishes the reconstruction property stated in Section 3.1. As for the security property stated in Section 3.1, this can be proved to hold modeling H_{G_0} as a random oracle, and assuming that the PVSS scheme is secure, and that the groups G and G^0 (with a pairing) satisfy a certain type of one-more Di e-Hellman hardness assumption, which can be stated as saying that no e cient adversary can win the following game with non-negligible probability:

注意，股份 xj 包含了 x 的 t-out-of-n Shamir 秘密共享。因此，出现在(1)中的 j 值只是拉格兰奇插值 coe cients。这就建立了第 3.1 节中描述的重建属性。至于第 3.1 节中所述的安全性，这可以证明建模 hg0 是一个随机预言，并且假设 PVSS 方案是安全的，群 g 和群 G0(有配对)满足某种类型的多 Di e-Hellman 硬度假设，这可以说明没有任何一个古老的对手能够以不可忽略的概率赢得下面的博弈:

The challenger chooses $_1; : : : ; _k \in Z_q$ and $_1; : : : ; ' \in Z_q$ at random, and gives $fg^{\ i}\ g^k_{i=1}$ and $f(g^0)^{\ j}\ g^{'}_{j=1}$ to the adversary.

挑战者随机选择 1; : : ; k2zq 和 1; : : ; '2zq，并给对手 fg i gki = 1 和 f (g0) j g' j = 1。

The adversary makes a sequence of queries to the challenger, each of which is a vector of the form $f_{i;j}g_{i;j}$, to which the challenger responds with

对手向挑战者提出一系列查询，每个查询都是挑战者回答的 f i; jgi; j 形式的向量

$$\prod_{i;j} (g^0)^{\ i\ j\ \ i;j} :$$

$$(g0)\ i\ j\ i; j:$$

To end the game, the adversary outputs a vector $f_{i;j}g_{i;j}$ and a group element $h^0 \in G^0$, and wins the game if

为了结束游戏，对手输出一个向量 f i; jgi; j 和一个组元素 h02 G0，如果

$$h^0 = \prod_{i;j} (g^0)^{\ i\ j\ \ i;j}$$

$$H0 = \prod_{i;j} (g0)\ i\ j\ i; j$$

and the output vector $f_{i;j}g_{i;j}$ is not a linear combination of the query vectors.

输出向量 f i; jgi; j 不是查询向量的线性组合。

While this type of one-more Di e-Hellman assumption is needed in the case where t > f + 1, one can get by with a weaker assumption when t = f + 1 (the so-called co-CDH assumption, on which the security of the ordinary BLS scheme is based).

虽然在 t > f + 1 的情况下需要这种类型的多 Di e-Hellman 假设，但当 t = f + 1 时，可以使用较弱的假设(所谓的 co-CDH 假设，普通 BLS 方案的安全性是基于此假设)。

3.6 A resharing protocol

3.6 再分享协议

The basic DKG protocol can be easily modi ed so that instead of creating a sharing of a fresh random secret x, it instead creates a fresh, random sharing of a previously shared secret.
基本的 DKG 协议可以很容易地进行修改，以便不创建一个新的随机秘密 x 的共享，而是创建一个新的、以前共享的秘密的随机共享。

- Step 1 of the basic protocol is modi ed so that each replica broadcasts a signed dealing of its existing share.

对基本协议的第一步进行了修改，以便每个副本都广播一个已签名的现有共享交易。

- Step 2 is modi ed so that a set of t valid signed dealings is agreed upon. Also, each

步骤 2 被修改，以便商定一组有效的签字交易

dealing is veri ed to ensure that it is indeed a dealing of the appropriate existing share (this means that the value of $A_{i;0}$ in the ith dealing should be equal to the old value of V_i).
交易须经核实，以确保交易的确是适当的现有份额(即第 1 次交易中的 Ai; 0 的价值应等于旧的 Vi 的价值)。

- In Step 3, the computation of the new x_j (and V_j) values weight the sum (and product) on i Lagrange interpolation coe cients.

在第三步中，新的 xj (和 Vj)值的计算对 i 拉格兰奇插值 cocients 上的和(和乘积)加权。

4 Peer-to-peer layer
点对点层

The peer-to-peer layer's task is to transport protocol messages between the replicas in a subnet. These protocol messages consist of
对等层的任务是在子网中的副本之间传输协议消息。这些协议消息包括

- messages used to implement consensus, e.g., block proposals, notarizations, etc. (see Section 5);
用以达成共识的讯息，例如: 阻挠建议书、公证等(见第 5 节) ;

- ingress messages (see Section 6).
输入信息(见第 6 节)。

Basically, the service provided by the peer-to-peer is a \best e ort" broadcast channel:
基本上，点对点提供的服务是一个最好的广播频道:

> if an honest replica broadcasts a messages, then that message will eventually be received by all honest replicas in the subnet.
> 如果一个诚实的副本广播一条消息，那么该消息最终将被子网中所有诚实的副本接收。

Design goals include the following:
设计目标包括:

- Bounded resources. All algorithms must work with bounded resources (memory, bandwidth, CPU).
有限资源。所有算法都必须使用有限资源(内存、带宽、 CPU)。

- Prioritization. Di erent messages may be treated with di erent priorities, depend-ing on certain attributes (e.g., type, size, round), and these priorities may change over time.
优先顺序。根据某些属性(例如，类型、大小、圆形) ，不同的消息可以使用不同的优先级进行处理，并且这些优先级可能随着时间的推移而改变。

- E ciency. High throughput is more important than low latency.
高吞吐量比低延迟更重要。

- DOS/SPAM resilience. Corrupt replicas should prevent honest replicas from com-municating with one another.
DOS/SPAM 弹性。腐败的副本应该阻止诚实的副本彼此通信。

Observe that in the consensus protocol, some messages, notably block proposals (which can be quite large), will be rebroadcast by all replicas. This is necessary to ensure correct behavior of that protocol. However, if implemented naively, this would be a huge waste of resources. To avoid having all replicas broadcasting the same message, the peer-to-peer layer makes use of an advertise-request-deliver mechanism. Instead of broadcasting a (large) message directly, it will instead broadcast a (small) advertisement for the message: if a replica receives such an advertisement, has not already received, and deems the message to be important, it will request that the message is delivered. This strategy decreases bandwidth utilization at the cost of higher latency. For small messages, this trade-o is not worthwhile, and it makes more sense to just send the message directly, rather than an advertisement.

请注意，在共识协议中，一些消息，尤其是阻止建议(可能相当大)，将被所有副本重新广播。这对于确保协议的正确行为是必要的。然而，如果天真地执行，这将是一个巨大的资源浪费。为了避免所有的副本都传播相同的信息，对等层使用了广告-请求-传递机制。它不直接广播(大)消息，而是为该消息广播(小)广告：如果副本收到这样的广告，但尚未收到，并认为该消息很重要，它将请求传递该消息。这种策略以更高的延迟为代价，降低了带宽利用率。对于小消息，这种交易是不值得的，直接发送消息比广告更有意义。

For relatively small subnets, a replica that wishes to broadcast a message will send an advertisement to all replicas in the subnet, each of which may then request that the message is delivered. For larger subnets, this advertise-request-deliver mechanism may operate over an overlay network. An overlay network is a connected, undirected graph whose vertices comprise the replicas in a subnet. Two replicas are peers if there is an edge connecting them in this graph, and replica only communicates with its peers. So when a replica wishes to broadcast a message, it sends an advertisement for that message to its peers. Those peers may request that the message be delivered, and upon receiving the message, if certain conditions are met, those peers will advertise the message to their peers. This is essentially a gossip network. This strategy again decreases bandwidth utilization at the cost of even higher latency.

对于相对较小的子网，希望广播消息的副本将向子网中的所有副本发送广告，然后每个副本可以请求传递消息。对于较大的子网，这种广告-请求-传递机制可以在覆盖网络上运行。覆盖网络是一个连接的无向图，其顶点包含子网中的副本。如果图中有一条边连接两个副本，那么两个副本就是对等点，并且副本只与它的对等点通信。因此，当一个副本希望广播一条消息时，它会向它的同伴发送该消息的广告。这些对等点可能请求传递消息，在收到消息时，如果满足某些条件，这些对等点将向它们的对等点发布消息。这本质上是一个八卦网络。这种策略又一次以更高的延迟为代价降低了带宽利用率。

5 Consensus Layer
共识层

The job of the consensus layer of the IC is to order inputs so that all replicas in a subnet will process such inputs in the same order. There are many protocols in the literature for this problem. The IC uses a new consensus protocol, which is described here at a high level. For more details, see the paper [CDH$^+$21] (in particular, Protocol ICC1 in that paper).
IC 的共识层的工作是对输入进行排序，这样子网中的所有副本将以相同的顺序处理这些输入。在这个问题的文献中有许多协议。集成电路使用一个新的共识协议，这里在高层描述。有关更多细节，请参见文件[CDH + 21](特别是该文件中的协议 ICC1)。

Any secure consensus protocol should guarantee two properties, which (roughly stated)
任何安全的协商一致协议都应该保证两个特性，这两个特性(粗略地说)
are:
是:

- safety: all replicas in fact agree on the same ordering of inputs, and
  安全性: 所有副本实际上都同意相同的输入顺序，并且

- liveness: all replicas should make steady progress.
  活性: 所有副本都应该取得稳定的进展。

The paper [CDH$^+$21] proves that the IC consensus protocol satis es both of these prop-erties
文章[CDH + 21]证明了 IC 协商一致协议满足这两个性能
The IC consensus protocol is designed to be
IC 共识协议的设计目的是

- extremely simple, and
  非常简单，而且

- robust: performance degrades gracefully when some replicas are malicious.
  : 当一些副本是恶意的时候，性能会优雅地下降。

5.1 Assumptions
5.1 假设

As discussed in the introduction, we assume
正如引言中所讨论的，我们假设

- a subnet of n replicas, and
  N 个副本的子网，以及

- at most f < n=3 of the replicas are faulty.
  最多 f < n = 3 个副本是错误的。

Faulty replicas may exhibit arbitrary, malicious (i.e., Byzantine) behavior.
错误的复制品可能表现出任意的、恶意的(即拜占庭式的)行为。

We assume that communication is asynchronous, with no a priori bound on the delay of messages sent between replicas. In fact, the scheduling of message delivery may be completely under adversarial control. The IC consensus protocol guarantees safety under this very weak communication assumption. However, to guarantee liveness, we need to

assume a form of partial synchrony, which (roughly stated) says that the network will be periodically synchronous for short intervals of time. In such intervals of synchrony, all undelivered messages will be delivered in less than time , for some xed bound . The bound does not have to be known in advance (the protocol is initialized with a reasonable bound, but will dynamically adapt and increase this bound if it is too small). Regardless of whether we are assuming an asynchronous or a partially synchronous network, we assume that every message sent from one honest replica to another will eventually be delivered.

我们假设通信是异步的，没有先验限制在副本之间发送消息的延迟。事实上，消息传递的调度可能完全处于对抗性的控制之下。IC 共识协议保证了在这种非常弱的通信假设下的安全性。然而，为了保证活性，我们需要假设一种部分同步的形式，这种形式(粗略地说)表示网络将在短时间间隔内周期性地同步。在这样的同步间隔中，所有未发送的消息都会在一定范围内以小于时间的速度发送。界限不必事先知道(协议初始化时有一个合理的界限，但是如果界限太小，则会动态地调整和增加该界限)。无论我们假设是异步网络还是部分同步网络，我们都假设从一个诚实副本发送到另一个诚实副本的每条消息最终都会被传递。

5.2 Protocol overview
5.2 议定书概述

Like a number of consensus protocols, the IC consensus protocol is based on a blockchain. As the protocol progresses, a tree of blocks is grown, starting from a special genesis block that is the root of the tree. Each non-genesis block in the tree contains (among other things) a payload, consisting of a sequence of inputs, and a hash of the block’s parent in the tree. The honest replicas have a consistent view of this tree: while each replica may

像许多共识协议一样，IC 共识协议是基于区块链的。随着协议的进展，一个块树开始生长，从一个特殊的起源块开始，这个起源块是树的根。树中的每个非起源块包含(除其他外)一个有效负载，由输入序列和树中块的父块的散列组成。诚实的副本对这个树有一个一致的视图：而每个副本可能

have a di erent, partial view of this tree, all the replicas have a view of the same tree. In addition, as the protocol progresses, there is always a path of nalized blocks in this tree. Again, the honest replicas have a consistent view of this path: while each replica may have a di erent, partial view of this path, all the replicas have a view of the same path. The inputs in the payloads of the blocks along this path are the ordered inputs will be processed by the execution layer of the Internet Computer (see Section 7).
这棵树有不同的部分视图，所有的复制品都有同一棵树的视图。此外，随着协议的进展，这棵树中总会有一条 nalized 块的路径。同样，诚实的副本具有这条路径的一致视图: 尽管每个副本可能具有这条路径的不同部分视图，但所有副本都具有相同路径的视图。沿着这条路径的块的有效负载中的输入是由 Internet 计算机的执行层处理的有序输入(参见第 7 节)。

The protocol proceeds in rounds. In round h of the protocol, one or more blocks of height h are added to the tree. That is, the blocks added in round h are always at a distance of exactly h from the root. In each round, a pseudo-random process is used to assign each replica a unique rank, which is an integer in the range 0; : : : ; n 1. This pseudo-random process is implemented using a random beacon (see Section 5.5 below). The replica of lowest rank is the leader of that round. When the leader is honest and the network is synchronous, the leader will propose a block, which will be added to the tree; moreover, this will be the only block added to the tree in this round and it will extend the nalized path. If the leader is not honest or the network is not synchronous, some other replicas of higher rank may also propose blocks, and also have their blocks added to the tree. In any case, the logic of the protocol gives highest priority to the leader's proposed block and some block or blocks will be added to this tree in this round. Even if the protocol proceeds for a few rounds without extending the nalized path, the height of the tree will continue to grow with each round, so that when the nalized path is extended in round h, the nalized path will be of length h. A consequence of this, even if the latency occasionally increases because of faulty replicas or unexpectedly high network latency, the throughput of the protocol remains essentially constant.

协议一轮一轮地执行。在协议的第一轮 h 中，一个或多个高度 h 块被添加到树中。也就是说，在第一轮 h 中添加的块始终与根 h 保持一定的距离。在每个回合中，使用伪随机过程为每个副本分配一个唯一的秩，即范围为 0: : : ; n1 的整数。这个伪随机过程是通过一个随机信标实现的(参见下面的 5.5 节)。最低等级的复制品是这一轮的领先者。当领导者诚实且网络是同步的时候，领导者会提出一个块，这个块将被添加到树中，而且这将是这一轮中唯一添加到树中的块，它将扩展 nalized 路径。如果领导者不诚实或网络不同步，其他一些级别较高的副本也可能提出块，并将它们的块添加到树中。在任何情况下，该协议的逻辑给予最高优先级的领导者提出的块，一些块或块将添加到该树在这一轮。即使协议在没有扩展 nalized 路径的情况下进行了几个回合，树的高度也会随着每个回合继续增长，因此当 nalized 路径在 h 回合中扩展时，nalized 路径的长度为 h。这样的结果是，即使由于错误的副本或意外的高延迟，延迟时间偶尔会增加，协议的吞吐量基本上保持不变。

5.3 Additional properties
5.3 附加属性

An additional property enjoyed by the IC consensus protocol (just like PBFT [CL99] and HotStu [AMN+20], and unlike others, such as Tendermint [BKM18]) is optimistic responsiveness [PS18], which means that when the leader is honest, the protocol may proceed at the pace of the actual network delay, rather than some upper bound on the network delay.

IC 协商一致协议(就像 PBFT [CL99]和 HotStu [AMN + 20] ，而不像其他协议，如 Tendermint [BKM18])的另一个特性是乐观响应性[PS18] ，这意味着当领导者诚实时，协议可以按照实际网络延迟的速度进行，而不是网络延迟的某个上限。

We note that the simple design of the IC consensus protocol also ensures that its performance degrades quite gracefully when and if Byzantine failures actually do occur. As pointed out in [CWA+09], much of the recent work on consensus has focused so much on improving the performance in the \optimistic case" where there are no failures, that the resulting protocols are dangerously fragile, and may become practically unusable when failures do occur. For example, [CWA+09] show that the throughput of existing implementations of PBFT drops to zero under certain types of (quite simple) Byzantine behavior. The paper [CWA+09] advocates for robust consensus, in which peak performance under optimal conditions is partially sacri ced in order to ensure reasonable performance when some parties actually are corrupt (but still assuming the network is synchronous). The IC consesus protocols is indeed robust in the sense of [CWA+09]: in any round where the leader is corrupt (which itself happens with probability less than 1=3), the protocol will e ectively allow another party to take over as leader for that round, with very little fuss, to move the protocol forward to the next round in a timely fashion.

我们注意到，IC 共识协议的简单设计也确保了当拜占庭故障确实发生时，其性能优雅地下降。正如在[CWA + 09]中指出的那样，最近关于协商一致的许多工作主要集中在改善乐观情况下的性能,”在这种情况下没有任何失败，由此产生的协议非常脆弱，而且在发生失败时可能实际上无法使用。例如，[CWA + 09]显示，在某些类型(非常简单)的 Byzantine 行为下，PBFT 的现有实现的吞吐量降至零。文件[CWA + 09]主张稳健的共识，其中最佳条件下的最高性能是部分牺牲，以确保合理的性能时，一些方面实际上是腐败的(但仍然假设网络是同步的)。IC 共识协议在[CWA + 09]的意义上确实是健壮的: 在任何一轮领导人腐败的情况下(这种情况发生的概率小于 1 = 3) ，该协议将有效地允许另一方接管该轮领导人的职位，而不会引起太大的麻烦，以便及时将协议推进到下一轮。

5.4 Public keys
5.4 公钥

To implement the protocol, each replica is associated with a public key for the BLS signature scheme [BLS01], and each replica also holds the corresponding secret signing key. The association of replicas to public keys is obtained from the registry maintained by the NNS (see Section 1.5). These BLS signatures will be used to authenticate messages sent by replicas.
为了实现该协议，每个副本都与 BLS 签名方案[BLS01]的公钥相关联，并且每个副本还持有相应的秘密签名密钥。副本与公钥的关联是从 NNS 维护的注册中心获得的(参见 1.5 节)。这些 BLS 签名将用于验证副本发送的消息。

The protocol also uses the signature aggregation feature of BLS signatures [BGLS03], which allows many signatures on the same message to be aggregated into a compact multi-signature. The protocol will use these multi-signatures for notarizations (see Section 5.7) and nalizations (see Section 5.8), which are aggregations of n f signatures on messages of a certain form.
该协议还使用了 BLS 签名的签名聚合特性[BGLS03]，它允许同一消息上的许多签名聚合成一个紧凑的多重签名。协议将使用这些多重签名进行公证(参见 5.7 节)和纳利化(参见 5.8 节)，它们是某种形式的消息上 n 个 f 签名的集合。

5.5 Random Beacon
5.5 随机信标

In addition to BLS signatures and multi-signatures as discussed above, the protocol makes use of a BLS threshold signature scheme to implement the above-mentioned random beacon. The random beacon for height h is a (f + 1)-threshold signature on a message unique to height h. In each round of the protocol, each replica broadcasts its share of the beacon for the next round, so that when the next round begins, all replicas should have enough shares to reconstruct the beacon for that round. As discussed above, the random beacon at height h is used to assign a pseudo-random rank to each replica that will be used in round h of the protocol. Because of the security properties of the threshold signature, an adversary will not be able to predict the ranking of the replicas more than one round in advance, and these rankings will e ectively be as good as random. See Section 3 for more on BLS threshold signatures.
除了上面讨论的 BLS 签名和多重签名外，该协议还利用 BLS 门限签名方案来实现上述随机信标。高度 h 的随机信标是高度 h 独有的消息上的(f + 1)-门限签名。在协议的每一轮中，每个副本广播它在下一轮中的信标份额，因此当下一轮开始时，所有副本都应该有足够的份额来重建该轮中的信标。如上所述，在高度 h 上的随机信标被用来为每个副本分配一个伪随机秩，这个伪随机秩将用于协议的第 h 轮。由于门限签名的安全性质，对手无法提前一轮以上预测副本的排名，这些排名实际上和随机排名一样好。有关 BLS 阈值签名的更多信息，请参见第 3 节。

5.6 Block making
5.6 砌块

Each replica may at di erent points in time play the role of a block maker. As a block maker in round h, the replica proposes a block B of height h that to be child of a block B^0

of height h 1 in the tree of blocks. To do this, the block maker rst gathers together a payload consisting of all inputs it knows about (but not including those already included in payloads in blocks in the path through the tree ending at B^0). The block B consists of
每个复制品可以在不同的时间点扮演一个砌块制作者的角色。作为圆 h 中的块制造者，副本提出了一个高度为 h 的块 b，它是块树中高度为 h1 的块 b0 的子块。为了做到这一点，块制造者首先收集一个有效负载，其中包含它所知道的所有输入(但不包括那些已经包含在有效负载中的输入，这些有效负载位于以 b0 结束的树的路径中的块)。块 b 由

- the payload,
有效载荷，

- the hash of B^0,
b0 的散列，

- the rank of the block maker,
砌块匠的级别，

- the height h of the block.
积木的高度。

After forming the block B, the block maker forms a block proposal, consisting of
在形成区块 b 之后，区块制造者形成一个区块建议，包括

- the block B,
B 区块，

- the block maker’s identity, and
制造商的身份，以及

- the block maker's signature on B.
砌块匠在 b 上的签名。

A block maker will broadcast its block proposal to all other replicas.
一个砌块制造商将向所有其他复制品广播他的砌块建议。

5.7 Notarization
5.7 公证

A block is e ectively added to the tree of blocks when it becomes notarized. For a block to become notarized, n f distinct replicas must support its notarization.
当一个块被公证后，它会被有效地添加到块树中。对于一个要公证的块，n 个不同的副本必须支持它的公证。

Given a proposed block B at height h, a replica will determine if the proposal is valid, which means that B has the syntactic form described above. In particular, B should contain the hash of a block B^0 of height h^0 that is already in the tree of blocks (i.e., already notarized). In addition, the payload of B must satisfy certain conditions (in particular, all of the inputs in the payload must satisfy various constraints, but these constraints are generally independent of the consensus protocol). Also, the rank of the block maker (as recorded in the block B) must match the rank assigned in round h by the random beacon to the replica that proposed the block (as recorded in the block proposal) .

给定一个高度为 h 的建议块 b，副本将确定该建议是否有效，这意味着 b 具有上述语法形式。特别地，b 应该包含一个已经在块树中的高度 h0 的 b0 块的散列(即已经公证过的)。此外，b 的有效载荷必须满足某些条件(特别是，有效载荷中的所有输入必须满足各种约束，但这些约束通常独立于协商一致协议)。此外，块制作者的等级(记录在块 b 中)必须与随机信标在第 h 轮中分配给提出块的副本的等级(记录在块建议中)相匹配。

If the block is valid and certain other constraints hold, the replica will support the notarization of the block by broadcasting a notarization share for B, consisting of
如果块有效且存在某些其他约束，则副本将通过广播 b 的公证共享来支持块的公证，其中包括

- the hash of B,
B 的散列，

- the height h of B,
B 的高度 h，

- the identity of the supporting replica, and
支持复制品的身份，以及

- the supporting replica's signature on a message comprising the hash of B and the height h.
支持副本的签名在包含 b 的散列和高度 h 的消息上。

Any set of n f notarization shares on B may be aggregated together to form a notarization for B, consisting of
B 上的任意一组 n f 公证股份可以聚合在一起形成 b 的公证，包括

- the hash of B,
B 的散列，

- the height h of B,

B 的高度 h,

- the set of identities of the n f supporting replicas,

支持复制的 n 个 f 的一组身份,

- an aggregation of the n f signatures on the message comprising the hash of B and the height h.

消息上 n 个 f 签名的集合，包括 b 的散列和 h 的高度。

As soon as a replica obtains a notarized block of height h, it will nish round h, and will subsequently not support the notarization of any other blocks at height h. At this point in time, such a replica will also relay this notarization to all other replicas. Note that this replica may have obtained the notarization either by (1) receiving it from another replica, or (2) aggregating n f notarization shares that it has received.

一旦一个副本获得一个经过公证的高度为 h 的块，它将在 h 周围结束，并且随后将不支持在高度为 h 的任何其他块的公证。在这个时候，这样的副本也会将这个公证传递给所有其他的副本。注意，这个副本可以通过(1)从另一个副本接收它，或者(2)聚合它接收的 n 个公证共享来获得公证。

The growth invariant states that each honest replica will eventually complete each round and start the next, so that the tree of notarized blocks continues to grow (and this holds only assuming asynchronous eventual delivery, and not partial synchrony). We prove the growth invariant below (see Section 5.11.4).

增长不变式指出，每个诚实的副本最终都会完成每一轮，并开始下一轮，这样公证块树就会继续增长(这只是假设异步的最终交付，而不是部分同步)。我们在下面证明了增长不变性(见第 5.11.4 节)。

5.8 Finalization
5.8 定稿

There may be more than one notarized block at a given height h. However, if a block is nalized, then we can be sure that there is no other notarized block at height h. Let us call this the safety invariant.
在给定的高度 h 上可能有不止一个经过公证的块。然而，如果一个块被归纳，那么我们可以确定在高度 h 上没有其他经过公证的块。

For a block to become nalized, n f distinct replicas must support its nalization. Recall that round h ends for a replica when it obtains a notarized block B of height h. At that point in time, such a replica will check if it supported the notarization of any block at height h other than block B (it may or may not have supported the notarization of B itself). If not, the replica will support the nalization of B by broadcasting a nalization share for B. A nalization share has exactly the same format as a notarization share (but is tagged in such a way notarization shares and nalization shares cannot be confused with one another). Any set of n f nalization shares on B may be aggregated together to form a nalization for B, which has exactly the same format as a notarization (but again, is appropriately tagged). Any replica that obtains a nalized block will broadcast the nalization to all other replicas.

对于要成为固定化的块，n 个不同的副本必须支持其固定化。回想一下，当一个副本获得一个经过公证的高度为 h 的块 b 时，它的圆 h 结束。此时，这样的副本将检查它是否支持除 b 块之外的任何高度 h 的块的公证(它可能支持也可能不支持 b 本身的公证)。如果没有，副本将通过广播 b 的 nalization 共享来支持 b 的 nalization 化。国有化共享与公证共享具有完全相同的格式(但是以这样一种方式进行标记，公证共享和国有化共享不能相互混淆)。B 上的任何 n 个 f 化共享集合可以聚合在一起形成 b 的 n 化，它具有与公证完全相同的格式(但同样，被适当地标记)。任何获得 nalized 块的副本都将向所有其他副本广播 nalization。

We prove the safety invariant below (see Section 5.11.5). One consequence of the safety invariant is the following. Suppose two blocks B and B^0 are nalized, where B has height h, B^0 has height h^0 h. Then the safety invariant implies that the path in the tree of notarized blocks ending at B^0 is a pre x of the path ending at B (if not, then there would be two notarized blocks at height h^0, contradicting the nalization invariant). Thus, whenever a replica sees a nalized block B, it may view all ancestors of B as being implicitly nalized, and because of the safety invariant, the safety property is guaranteed to hold for these (explicitly and implicitly) nalized blocks | that is, all replicas agree on the ordering of these nalized blocks.

我们证明了下面的安全不变量(见第 5.11.5 节)。安全不变量的一个结果如下。假设两个区块 b 和 b0 被定义，其中 b 有高度 h，b0 有高度 h。那么安全不变量意味着以 b0 结束的公证块树中的路径是以 b 结束的路径的前 x (如果不是，那么在 h0 高度将有两个公证块，与 nalization 不变量相矛盾)。因此，当一个副本看到一个 nalized 块 b 时，它可以将 b 的所有祖先看作是隐式 nalized 的，并且由于安全不变性，保证了这些(显式和隐式) nalized 块 | 的安全性，也就是说，所有副本都同意这些 nalized 块的顺序。

5.9 Delay functions
5.9 延迟功能

The protocol makes use of two delay functions, $_m$ and $_n$, which control the timing of block making and notarization activity. Both of these functions map the rank r of the

proposing replica to a nonnegative delay amount, and it is assumed that each function is monotonely increasing in r, and that $_m$(r) $_n$(r) for all r = 0; : : : ; n 1. The recommended de nition of these functions is $_m$(r) = 2 r and $_n$(r) = 2 r + , where is an upper bound on the time to deliver messages from one honest replica to another, and
该协议使用了两个延迟函数 m 和 n，它们控制块制作和公证活动的时间。这两个函数都将提出的副本的秩 r 映射到一个非负延迟量，并且假设每个函数在 r 中是单调增加的，并且对于所有 r = 0 的 m (r) n (r) ; : : ; n1。这些函数的推荐定义是 m (r) = 2r，n (r) = 2r + ，这是将消息从一个诚实副本传递到另一个诚实副本的时间上限

0 is a \governor" to keep the protocol from running too fast. With these de nitions, liveness will be ensured in those rounds in which (1) the leader is honest, and (2) messages really are delivered between honest replicas within time . Indeed, if (1) and (2) both hold in a given round, then the block proposed by the leader in that round will be nalized. Let us call this the liveness invariant. We prove this below (see Section 5.11.6).

0 是一个调控器”，以防止协议运行过快。有了这些定义，在(1)领导者是诚实的，以及(2)信息在时间内真正在诚实的副本之间传递的那些回合中，活力将得到保证。实际上，如果(1)和(2)在一个给定的回合中都保持不变，那么该回合中领导者提出的方案就会被消除。让我们称之为活性不变量。我们在下面证明了这一点(见第 5.11.6 节)。

5.10 An example
5.10 一个例子

Figure 2 illustrates a block tree. Each block is labeled with its height (30, 31, 32, . . .) and the rank of its block maker. The gure also shows that each block in the tree is notarized, as indicated by the $_N$ symbol. This means that for each notarized block in the tree, at least n f distinct replicas supported its notarization. As one can see, there can be more
图 2 演示了一个块树。每个块都标有它的高度(30,31,32，...)和块制作者的等级。图表还显示树中的每个块都经过公证，如 n 个符号所示。这意味着对于树中的每个经过公证的块，至少有 n 个不同的副本支持其公证。正如我们所看到的，还有更多

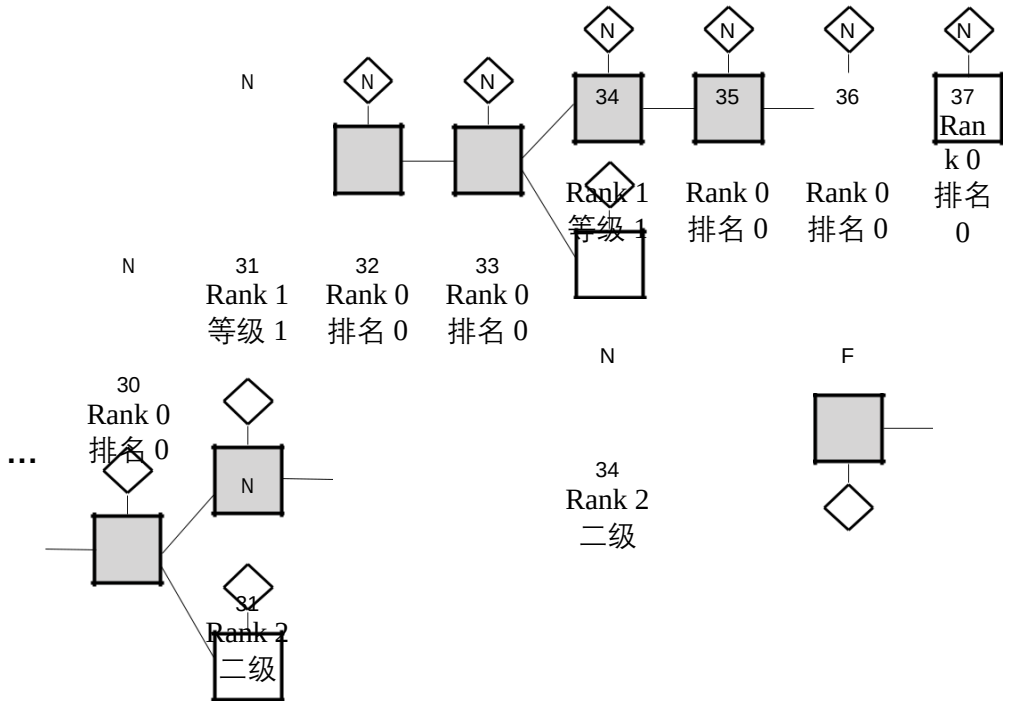

Figure 2: An example tree of blocks
图 2: 一个块的示例树

than one notarized block in the tree at a given height. For example, at height 32, we see there are two notarized blocks, one proposed by block makers of rank 1 and 2. The same thing happens at height 34. We can also see that the block at height 36 is also explicitly nalized, as indicated by the F symbol. This means that n f distinct replicas supported this block's nalization, which means that these replicas (or at least, the honest replicas among these) did not support the notarization of any other block. All of the ancestors of this block, which are shaded gray, are considered implicitly nalized.
在树的一个给定高度的公证块。例如，在 32 的高度，我们看到有两个公证块，其中一个是由级别为 1 和 2 的块制作者提出的。同样的事情发生在 34 高度。我们还可以看到 36 高度的块也被显式地 nalized 了，如 f 符号所示。这意味着 n 个不同的副本支持这个块的公证化，这意味着这些副本(或者至少是其中真实的副本)不支持任何其他块的公证。这个块的所有祖先都是灰色阴影，被认为是隐式的。

5.11 Putting it all together
5.11 把它们放在一起

We now describe in more detail how the protocol works; speci cally, we describe more precisely when a replica will propose a block and when a replica will support the notarization of a block. A given replica P will record the time at which it enters a given round h, which happens when it has obtained (1) some notarization for a block of height h 1, and (2) the random beacon for round h. Since the random beacon for round h has been determined, P can determine its own rank r_P , as well as the rank r_Q of each other replica Q for round h.

现在我们更详细地描述协议是如何工作的；具体地说，我们更精确地描述了副本何时提出块，以及副本何时支持块的公证。一个给定的副本 p 将记录它进入一个给定的轮 h 的时间，这个时间发生在它获得了(1)某个高度为 h1 的块的公证，和(2)该轮 h 的随机信标。由于圆 h 的随机信标已经确定，p 可以确定自己的等级 rP，以及圆 h 的相互副本 q 的等级 rQ。

5.11.1 Random beacon details
5.11.1 随机信标细节

As soon as a replica has received the random beacon for round h, or enough shares to contruct the random beacon for round h, it will relay the random beacom for round h to all other replicas. As soon as a replica enters round h, it will generate and broadcast its share of the random beacon at round h + 1.
一旦一个副本接收到第 h 轮的随机信标，或者有足够的份额构造第 h 轮的随机信标，它就会将第 h 轮的随机信标中继到所有其他副本。一旦一个副本进入 h 轮，它将在 h + 1 轮生成并广播它所分享的随机信标。

5.11.2 Block making details
5.11.2 Block 制作细节

Replica P will only propose its own block B_P provided (1) at least $_m(r_P)$ time units have passed since the beginning of the round, and (2) there is no valid lower ranked block currently seen by P .
Replica p 将只提出它自己的块 BP，条件是(1)至少 m (rP)时间单位自回合开始以来已经过去，以及(2)目前 p 看到的没有有效的低排序块。

Note that since P is guaranteed to have a notarized block of height h 1 when it enters round h, it can make its proposed block a child of this notarized block (or any other notarized block of height h 1 that it may have). Also note that when p broadcasts its proposal for B_P , it must also ensure that it also has relayed the notarization of B_P 's parent to all replicas.

注意，由于 p 在进入第一轮 h 时保证有一个高度为 h1 的公证块，因此它可以使其提议的块成为这个公证块(或者它可能拥有的任何其他高度为 h1 的公证块)的子块。还需要注意的是，当 p 广播 BP 的建议时，它必须确保它也已经将 BP 的父类的公证转发给了所有的副本。

Suppose a replica Q sees a valid block proposal from a replica P of rank $r_P < r_Q$ such that (1) at least $_m(r_P)$ time units have passed since the beginning of the round, and (2) there is no block of rank less than r_P currently seen by Q. Then at this point in time, if it has not already done so, Q will relay this block proposal (along with the notarization of the proposed block's parent) to all other replicas.

假设一个副本 q 从秩 rP < rQ 的副本 p 中看到一个有效的块提案，使得(1)至少 m (rP)时间单位已经从这个回合的开始过去了，并且(2)没有一个秩小于 q 当前看到的 rP 的块。然后在这个时候，如果它还没有这样做，q 将中继这个块提议(连同提议块的父代的公证)到所有其他副本。

5.11.3 Notarization details
5.11.3 公证详情

Replica P will support the notarization of a valid block B_Q proposed by a replica Q of rank r_Q provided (1) at least $_n(r_Q)$ time units have passed since the beginning of the round, and (2) there is no block of rank less than r_Q currently seen by P .

Replica p 将支持由秩 rQ 的副本 q 提出的有效块 BQ 的公证，条件是(1)至少有 n 个(rQ)时间单位自回合开始以来已经过去，并且(2)没有秩小于目前 p 所看到的秩的块。

5.11.4 Proof of growth invariant
5.11.4 生长不变性的证明

The growth invariant states that each honest replica will eventually complete each round and start the next. Assume that all honest replicas have started round h. Let r be the rank of the lowest ranked honest replica P in round h. Eventually, P will either (1) propose its own block, or (2) relay a valid block proposed by a lower ranked replica. In either case, some block must eventually be supported by all honest replicas, which means that some block will become notarized and all honest replicas will nish round h. All honest replicas will also receive the shares needed to construct the random beacon for round h + 1, and so will start round h + 1.

增长不变性指出每个诚实的副本最终将完成每一轮，并开始下一轮。假设所有诚实副本都在第 h 轮开始，设 r 是第 h 轮排名最低的诚实副本 p 的排名。最终，p 要么(1)提出自己的块，要么(2)中继由排名较低的副本提出的有效块。在任何一种情况下，某个块最终必须由所有诚实的副本支持，这意味着某个块将被公证，而所有诚实的副本将在 h 周期结束。所有诚实的副本也将收到构建 h + 1 周期的随机信标所需的份额，因此将在 h + 1 周期开始。

5.11.5 Proof of safety invariant
5.11.5 安全不变量的证明

The safety invariant states that if a block is nalized in a given round, then no other block may be notarized in that round. Here is a proof of the safety invariant:
安全不变性指出，如果一个块在给定的一轮中被固定，那么在该轮中没有其他块可以被公证。下面是安全不变量的一个证明:

1. Suppose that the number of corrupt replicas is exactly f f < n=3.
假设损坏的副本数正好是 f f < n = 3。

2. If a block B is nalized, then its nalization must have been supported by a set S of at least n f f honest replicas (by the security property for aggregate signatures).
如果一个块 b 是固定化的，那么它的固定化必须得到至少 n f 个真实副本的集合 s 的支持(通过聚合签名的安全属性)。

3. Suppose (by way of contradiction) that another block B^0 =6 B were notarized. Then its notarization must have been supported by a set S^0 of at least n f f honest replicas (again, by the security property for aggregate signatures).
3.假设(通过矛盾的方式)另一块 B0 = 6b 被公证。然后，它的公证必须由至少 n f 诚实副本的集合 s0 支持(同样，由聚合签名的安全属性支持)。

4. The sets S and S^0 are disjoint (by the nalization logic).
集合 s 和 s0 是不相交的(通过归化逻辑)。

5. Therefore, n f jS [S^0j = jSj + jS^0j 2(n f f), which implies n 3f, a contradiction.
5.因此，n f jS [S0j = jSj + js0j2(n f f)，这意味着 n 3f，一个矛盾。

5.11.6 Proof of liveness invariant
5.11.6 活性不变量的证明

We say that the network is -synchronous at time t if all messages that have been sent by honest replicas at or before time t arrive at their destinations before time t.
如果在时间 t 之前或之前由诚实副本发送的所有消息在时间 t 之前到达目的地，则称网络在时间 t 之前是-同步的。

The liveness invariant may be stated as follows. Suppose that $_{n}(1)$ $_{m}(0) + 2$. Also suppose that in a given round h, we have
活性不变量可以说明如下。假设 n (1) m (0) + 2。还假设在给定的一轮 h 中，我们有

- the leader P in round h is honest,
第一轮中的 p 是诚实的,

- the rst honest replica Q to enter round h does so at time t, and
进入第一轮 h 的第一个诚实副本 q 在时间 t 时这样做，并且

- the network is -synchronous at times t and t + + $_{m}(0)$.
网络在 t 和 t + + m (0)时是-同步的。

Then the block proposed by P in round h will be nalized.
然后将 p 在第一轮 h 中提出的分块进行优化。

Here is a proof of the liveness invariant:
下面是一个活性不变量的证明:

1. Under partial synchrony at time t, all honest replicas will enter round h before time t + (the notarization that ended round h 1 for Q as well as the random beacon for round h random will arrive at all honest replicas before this time).
1.在时间 t 部分同步的情况下，所有诚实的副本将在时间 t + 之前进入第一轮 h (对 q 结束第一轮 h 的公证以及对第一轮 h 随机的随机信标将在此之前到达所有诚实的副本)。

2. The leader P in round h will propose a block B before time t + + $_{m}(0)$, and again
第一轮中的 p 将在 t + + m (0)之前提出一个区块 b，然后再次提出
by partial synchrony, this block proposal will be delivered to all other replicas before time t + 2 + $_{m}(0)$.
通过部分同步，这个块建议将在时间 t + 2 + m (0)之前交付给所有其他副本。

3. Since $_{n}(1)$ $_{m}(0) + 2$, the protocol logic guarantees that each honest replica supports the notarization of block B and no other block, and thus B will become notarized and nalized.
由于 n (1) m (0) + 2，协议逻辑保证每个诚实的副本支持块 b 的公证，而不支持其他块，因此 b 将成为公证和 nalized。

5.12 Other issues
5.12 其他问题

5.12.1 Growth latency
5.12.1 成长潜伏期

Under a partial synchrony assumption, we can also formulate and prove a quantitative version of the growth invariant. For simplicity, assume that the delay functions are de

ned as recommended above: $_m$(r) = 2 r and $_n$(r) = 2 r + , and further assume that . Suppose that at time t, the highest numbered round entered by any honest replica is h. Let r be the rank of the lowest ranked honest replica P in round h. Finally, suppose that the network is -synchronous at all times in the interval [t; t + (3r + 2)]. Then all honest replicas will start round h + 1 before time t + 3(r + 1) .
在部分同步假设下，我们也可以公式化和证明增长不变量的一个定量版本。为了简单起见，假设延迟函数是按照上面的建议设计的: m (r) = 2r，n (r) = 2r + ，并进一步假设。假设在时间 t 时，任何诚实副本输入的最高编号轮是 h。设 r 是在第 h 轮中排名最低的诚实副本 p 的秩。然后所有诚实的副本将在 t + 3(r + 1)之前的 h + 1 开始。

5.12.2 Locally adjusted delay functions
5.12.2 本地调整延迟功能

When a replica does not see any nalized blocks for several rounds, it will start increasing its own delay function $_n$ for notarization. Replicas need not agree on these locally adjusted notarization delay functions.
当一个副本在几个回合中没有看到任何纳化块时，它将开始增加自己的延迟函数 n 以进行公证。Replicas 不需要在这些本地调整的公证延迟函数上达成一致。

Also, while replicas do not explicitly adjust the delay function $_p$, we can mathematically model local clock drift by locally adjusting both delay functions.
此外，虽然副本不显式调整延迟函数 p，我们可以通过局部调整两个延迟函数来数学模型本地时钟漂移。

Thus, there are many delay functions, parameterized by replica and round. The critical condition $_n$(1) $_m$(0) + 2 needed for liveness then becomes max $_n$(1) min $_m$(0) +
因此，有许多延迟函数，参数化的副本和轮。活性所需的临界条件 n (1) m (0) + 2 然后变成最大 n (1) min m (0) +

2 , where the max and min are taken over all the honest replicas in a given round. Thus, if nalization fails for enough rounds, all honest replicas will eventually increase their notarization delay until this holds and nalization will then resume. If some honest replicas increase their notarization latency function more than other replicas, there is no penalty in terms of liveness (but there may be in terms of growth latency).
2，其中最大值和最小值取决于给定一轮中所有诚实的副本。因此，如果 nalization 在足够的回合中失败，那么所有诚实的副本最终都会增加它们的非 tarization 延迟，直到这种情况持续下去，然后 nalization 就会恢复。如果一些诚实的副本比其他副本更多地增加了公证延迟功能，那么在活性方面就不会受到惩罚(但在增长延迟方面可能会受到惩罚)。

5.12.3 Fairness
5.12.3 公平

Another property that is important in consensus protocols is fairness. Rather than give a general de nition, we simply observe that the liveness invariant also implies a useful fairness property. Recall that the liveness invariant basically says that in any round where the leader is honest and the network is synchronous, then the block proposed by the leader will be nalized. In those rounds where this happens, the fact that the leader is honest ensures that it will include in the payload of its block all of the inputs it knows about (modulo limits on the payload size). So, very roughly speaking, any input that is disseminated to enough replicas will be included in a nalized block in a reasonable amount of time with high probability.
协商一致协议的另一个重要特性是公平性。我们没有给出一个一般的定义，我们只是观察到活性不变量也暗示了一个有用的公平性属性。回想一下，活性不变量基本上是指在任何领导者诚实、网络同步的情况下，由领导者提出的块将被消除。在发生这种情况的那些回合中，领导者是诚实的这一事实确保了它将在其块的有效载荷中包含它所知道的所有输入(有效载荷大小的模数限制)。所以，粗略地说，任何被传播到足够多的副本的输入都会在合理的时间内以很高的概率包含在一个 nalized 块中。

6 Message Routing Layer
消息路由层

As discussed in Section 1.7, basic computational unit in the IC is called a canister, which is roughly the same as the notion of a process, in that it comprises both a program and its state. The IC provides a run-time environment for executing programs in a canister, and to communicate with other canisters and external users (via message passing).
如第 1.7 节所讨论的，集成电路中的基本计算单元称为“容器”，它与进程的概念大致相同，因为它包括程序及其状态。集成电路提供了一个运行时环境，用于在一个容器中执行程序，并与其他容器和外部用户通信(通过消息传递)。

The consensus layer (see Section 5) bundles inputs into payloads, which get placed into blocks, and as blocks are nalized, the corresponding payloads are delivered to the message routing layer, then processed by the execution environment, which updates the state of the canisters on the replicated state machine and generates outputs, and these outputs are processed by the message routing layer.

共识层(参见第 5 节)将输入捆绑到有效负载中，这些有效负载被放入块中，当块被固定化时，相应的有效负载被交付到消息路由层，然后由执行环境处理，执行环境更新复制状态机上的罐状态并生成输出，这些输出由消息路由层处理。

It is useful to distinguish between two types of inputs:
区分两种类型的输入是有用的:

ingress messages: these are messages from external users;
输入消息: 这些是来自外部用户的消息;

cross-subnet messages: these are messages from canisters on other subnets.
跨子网消息: 这些消息来自其他子网上的罐子。

We can also distinguish between two types of outputs:
我们还可以区分两种类型的输出:

ingress message responses: these are responses to ingress messages (which may be re-trieved by external users);
入口消息响应: 这些是对入口消息的响应(外部用户可以重新检索这些消息) ;

cross-subnet messages: these are messages to canisters on other subnets.
跨子网消息: 这些是其他子网上的邮件。

Upon receiving a payload from consensus, the inputs in that payload are placed into various input queues. For each canister C running on a subnet, there are several in-put queues: one for ingress messages to C, and for each other canister C^0 with whom C communicates, one for cross-subnet messages to C from C^0.

从协商一致接收有效负载后，该有效负载中的输入被放置到各种输入队列中。对于在子网上运行的每个罐 c，有几个输入队列: 一个用于向 c 输入消息，另一个用于与 c 通信的其他罐 C0，一个用于跨子网消息从 c0 到 c。

As described below in more detail, in a each round, the execution layer will consume some of the inputs in these queues, update the replicated state of the relevant canisters, and

正如下面更详细地描述的那样，在每一轮中，执行层将消耗这些队列中的一些输入，更新相关罐的复制状态，并且

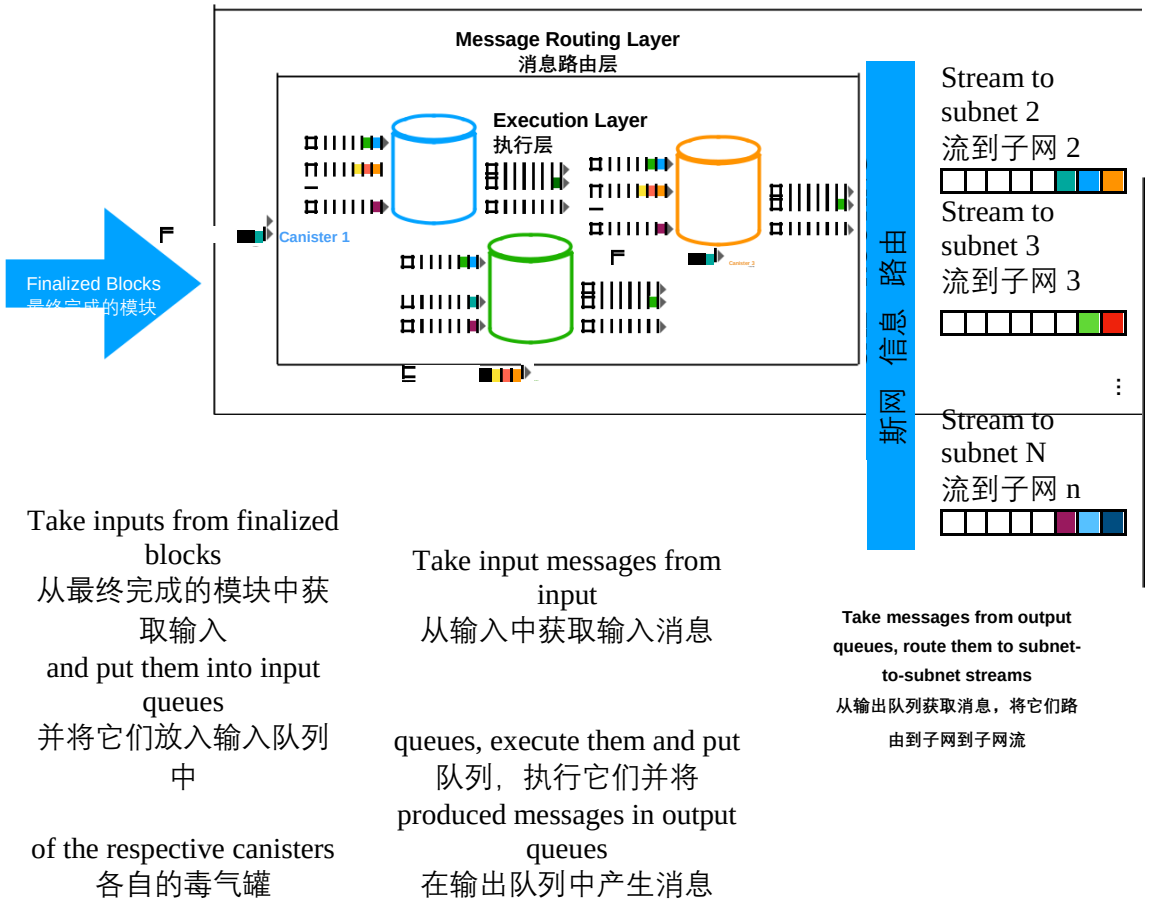

Figure 3: Message routing and execution layers
图 3: 消息路由和执行层

place outputs in various output queues. For each canister C running on a subnet, there are several output queues: for each other canister C^0 with whom C communicates, one for cross-subnet messages to C^0 from C. The message routing layer will take the messages in these output queues and place them into subnet-to-subnet streams to be processed by an crossnet transfer protocol, whose job it is to actually transport these messages to other subnets.
将输出放在不同的输出队列中。对于在子网上运行的每个罐 c，有几个输出队列：对于 c 与之通信的每个罐 C0，一个用于跨子网消息从 c 发送到 C0。消息路由层将获取这些输出队列中的消息，并将它们放入子网到子网的流中，由交叉网传输协议处理，交叉网传输协议的工作实际上是将这些消息传输到其他子网。

In addition to these output queues, there is also an ingress history data structure. Once an ingress message has been processed by a canister, a response to that ingress message will be recorded in this data structure. At that point, the external user who provided the ingress message will be able to retrieve the corresponding response. (Note that ingress history does not maintain the full history of all ingress messages.)
除了这些输出队列，还有一个入口历史记录数据结构。一旦进入消息被一个小罐处理，对该进入消息的响应将被记录在这个数据结构中。在这一点上，提供进入消息的外部用户将能够检索相应的响应。(请注意，入口历史并不保存所有入口消息的完整历史

We also should mention that in addition to cross-subnet messages, there are also intra-subnet messages, which are messages from one canister to another on the same subnet. The message routing layer moves such messages directly from output queues to correspond-ing input queues.

我们还应该提到，除了跨子网消息之外，还有子网内部消息，即同一子网上从一个容器到另一个容器的消息。消息路由层将这些消息直接从输出队列移动到相应的输入队列。

Figure 3 illustrates the basic functionality of the message routing and execution layers. Note that the replicated state comprises the the state of the canisters, as well as \system state", including the above-mentioned queues and streams, as well as the ingress history data structure. Thus, both the message routing and execution layers are involved in updating and maintaining the replicated state of a subnet. It is essential that all of this state is updated in a completely deterministic fashion, so that all replicas maintain exactly the same state.

图 3 说明了消息路由和执行层的基本功能。请注意，复制的状态包括罐的状态以及系统状态”，包括上述队列和流以及入口历史数据结构。因此，消息路由和执行层都涉及到更新和维护子网的复制状态。所有这些状态都必须以一种完全确定的方式更新，这样所有的副本都能精确地维护同一状态。

Also note that the consensus layer is decoupled from the message routing and execution layers, in the sense that any forks in the consensus blockchain are resolved before their payloads are passed to message routing, and in fact, consensus does not have to keep in

还要注意，共识层与消息路由和执行层是分离的，因为共识区块链中的任何分叉在其有效负载被传递到消息路由之前就已经被解析了，而且实际上，共识并不需要保留

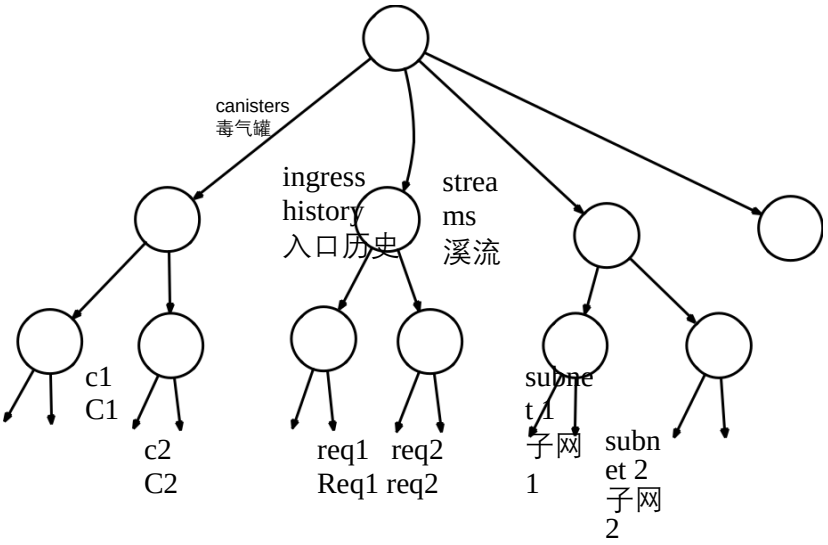

Figure 4: Per-round certi ed state organized as a tree
图 4: 每轮认证状态组织为一棵树

lock step with message routing and consensus and is allowed to run a bit ahead.
使用消息路由和共识锁定步骤，并允许提前一点运行。

6.1 Per-round certi ed state
每轮认证状态 6.1

In each round, some of the state of a subnet will be certi ed. The per-round certi ed state is certi ed using chain-key cryptography (see Section 1.6), speci cally, using the (n f)-out-of-n threshold signature scheme mentioned in Section 3. In more detail, after each replica generates the per-round certi ed state for a given round, it will generate a share of the corresponding threshold signature and broadcast this to all other replicas in its subnet. Upon collecting n f such shares, each replica can construct the resulting threshold signature, which serves as the certi cate for the per-round certi ed state for that round. Note that before signing, the per-round certi ed state is hashed as a Merkle tree [Mer87].
在每一轮中，子网的某些状态将被确认。每轮验证状态使用链密钥加密(参见 1.6 节)进行验证，特别是使用第 3 节中提到的(n f)-out-of-n 门限签名方案。更详细地说，在每个副本为给定的一轮生成每轮验证状态之后，它将生成相应的阈值签名的一部分，并将其广播到其子网中的所有其他副本。在收集 n 个这样的共享之后，每个副本可以构造最终的阈值签名，作为该轮每轮认证状态的证书。注意，在签名之前，每轮认证状态被散列为 Merkle 树[Mer87]。

The per-round certi ed state in a given round consists of
给定一轮中的每轮认证状态包括

1. cross-subnet messages that were recently added to the subnet-to-subnet streams;
最近添加到子网到子网流中的跨子网消息;

2. other metadata, including the ingress history data structure;
其他元数据，包括入口历史数据结构;

3. the Merkle-tree root hash of the per-round certi ed state from the previous round.
上一轮每轮确认状态的 Merkle-tree 根散列。

Note that the per-round certi ed state does not include the entire replicated state of a subnet, as this in general will be quite huge and it would be impractical to certify all of this state in every round.[6]
请注意，每轮认证的状态不包括子网的整个复制状态，因为这通常是相当巨大的，每轮认证所有这些状态是不切实际的

Figure 4 illustrates how the per-round certi ed state may be organized into a tree. The rst branch of the tree stores various metadata about each canister (but not the entire replicated state of the canister). The second branch stores the ingress history data struc-ture. The third branch stores information about the subnet-to-subnet streams, including a \window" of recently added cross-subnet messages for each stream. The other branches store other types of metadata, not discussed here. This tree structure may then be hashed into a Merkle tree, which has essentially the same size and shape as this tree.

图 4 说明了如何将每轮认证状态组织到树中。树的第一个分支存储关于每个罐子的各种元数据(但不是罐子的整个复制状态)。第二个分支存储入口历史数据结构。第三个分支存储子网到子网流的信息，包括最近添加的每个流的跨子网消息窗口。其他分支存储其他类型的元数据，这里没有讨论。这个树结构可能会被散列成一棵 Merkle 树，它的大小和形状基本上和这棵树一样。

[6] But see Section 8.2
但参见第 8.2 节

Per-round certi ed state is used in several ways in the IC:
每轮认证状态在集成电路中有几种用法:

- Output authentication. Cross-subnet messages and responses to ingress messages are authenticated using per-round certi ed state. Using the Merkle tree structure, an individual output (cross-subnet message or ingress message response) may be authen-ticated to any party by providing a threshold signature on the root of the Merkle tree, along with hash values on (and adjacent to) the path in the Merkle from the root to the leaf representing that output. The number if hash values needed to authenticate an individual output is therefore proportional to the depth of the Merkle tree, which is typically quite small, even if the size of the Merkle tree is very large. Thus, a single threshold signature can be used to e ciently authenticate many individual outputs.

输出认证。跨子网消息和对入口消息的响应使用每轮认证状态进行认证。使用 Merkle 树结构，一个单独的输出(跨子网消息或入口消息响应)可以通过在 Merkle 树的根上提供一个阈值签名，以及 Merkle 中从根到叶表示输出的路径上(和邻近的)的哈希值来对任何一方进行认证。因此，验证单个输出所需的散列值的数量与 Merkle 树的深度成正比，即使 Merkle 树的大小非常大，Merkle 树的深度通常也非常小。因此，一个单一的阈值签名可以用来验证许多单独的输出。

- Preventing and detecting non-determinism. Consensus guarantees that each replica processes inputs in the same order. Since each replica processes these in-puts deterministically, each replica should obtain the same state. However, the IC is designed with an extra layer of robustness to prevent and detect any (accidental) non-deterministic computation, should it arise. The per-round certi ed state is one of

防止和检测不确定性。Consensus 保证每个副本以相同的顺序处理输入。由于每个副本确定性地处理这些输入输入，每个副本应该获得相同的状态。然而，集成电路设计了一个额外的健壮性层，以防止和检测任何(偶然的)非确定性计算，如果它出现。每轮的确认状态是

the mechanisms used to do this. Since we use an (n f)-out-of-n threshold signature for certi cation, and since f < n=3, there can only be one sequence of states that is certi ed.
用来做这个的机械装置。由于我们使用一个(n f)-out-of-n 阈值签名进行验证，并且由于 f < n = 3，因此只能有一个状态序列被验证。

To see why state chaining is important, consider the following example. Suppose we have 4 replicas, P_1; P_2; P_3; P_4, and one is corrupt, say P_4. Each of the replicas P_1; P_2; P_3 start out in the same state.
为了了解为什么状态链接是重要的，考虑下面的例子。假设我们有 4 个副本，P1; P2; P3; P4，其中一个被破坏了，比如 P4。每个副本 P1; P2; p3 以相同的状态开始。

{ In round 1, because of a non-deterministic computation, P_1; P_2 compute a mes-sage m_1 to send to subnet A, while P_2 computes a message m'_1 to send to subnet A.
{在第 1 轮中，由于计算的不确定性，P1; p2 计算一个消息 m1 发送到子网 a，而 p2 计算一个消息 m01 发送到子网 a。

{ In round 2, P_1; P_3 compute a message m_2 to send to subnet B, while P_2 computes a message m'_2 to send to subnet B.

{在第二轮，P1; p3 计算一条消息 m2 发送到子网 b，而 p2 计算一条消息 m02 发送到子网 b。

{ In round 3, P_2; P_3 compute a message m_3 to send to subnet C, while P_2 computes a message m'_3 to send to subnet C.

{在第 3 轮中，P2; p3 计算一条消息 m3 发送到子网 c，而 p2 计算一条消息 m03 发送到子网 c。

This is illustrated in the following table:

如下表所示:

P_1	m_1 ! A	m_2 ! B	m'_3 ! C
P1 m1! a m2! b m03! c			
P_2	m_1 ! A	m'_2 ! B	m_3 ! C
P2 m1! a m02! b m3! c			
P_3	m'_1 ! A	m_2 ! B	m_3 ! C
P3 m01! a m2! b m3! c			

We are assuming that replicas P_1, P_2, and P_3 each individually perform a valid sequence of computations, but that because of non-determinism, these sequences are not identical. (Even though there is not supposed to be any non-determinism, in this example, we are supposing that there is.)

我们假设副本 P1、 p2 和 p3 各自执行一个有效的计算序列，但是由于不确定性，这些序列是不相同的。(即使不应该有任何非决定论，在这个例子中，我们假设有

Now suppose we did not chain the states. Because P_4 is corrupt and may sign anything, he could create a 3-out-of-4 signature on a round-1 state that says \m_1 ! A", and

现在假设我们没有锁住这些州。因为 p4 是腐败的，可以签署任何东西，他可以创建一个 3-out-4 签名在第一轮状态，说明 m1！A”和

similarly on a round-2 state that says \$m_2$! B", and on a round-3 state that says \$m_3$! C", even though the corresponding sequence
同样的，第二回合的状态是 m2！B”，第三轮状态为 m3！C”，即使相应的序列

$$m_1 ! A; m_2 ! B; m_3 ! C$$
M1! a; m2! b; m3! c

may not compatible with any valid sequence of computations. Worse yet, such an invalid sequence of computations could then lead to inconsistent states on other sub-nets.
可能与任何有效的计算序列不兼容。更糟糕的是，这样一个无效的计算序列可能会导致其他子网的不一致状态。

By chaining, we ensure that even if there is some non-determinism, any sequence of certi ed states corresponds to some valid sequence of computations that was actually carried out by honest replicas.
通过链接，我们确保即使存在一些非确定性，任何已验证的状态序列对应于一些有效的计算序列，这些计算实际上是由诚实的副本执行的。

- Coordination with consensus. The per-round certi ed state is also used to coordinate the execution and consensus layers, in two di erent ways:
协调一致。每轮认证状态也被用于协调执行和共识层面，通过两种不同的方式:

  { Consensus throttling. Each replica will keep track the latest round for which it is has a certi ed state | this is called the certi ed height. It will also keep track of the latest round for which it is has a notarized block | this is called the notarized height. If the notarized height is signi cantly greater than the certi ed height, this is a signal that execution is lagging consensus, and that consensus needs to be throttled. This lagging could be due to non-deterministic computation, or it could just be due to a more benign performance mismatch between the layers. Consensus is throttled by means of the delay functions discussed in Section 5.9 | speci cally, each replica will increase the \governor" value as the gap between notarized height and certi ed height grows (this makes use of the notion of \locally adjusted delay functions, as in Section 5.12.2).
  共识节流。每个副本都会跟踪最新一轮的状态，这个状态被称为确认高度。它也会记录最新的一轮，它有一个公证块 | 这被称为公证高度。如果公证的高度明显大于认证的高度，这是执行滞后于共识的信号，共识需要被扼杀。这种滞后可能是由于计算的不确定性，也可能仅仅是由于层之间的良性性能不匹配。Consensus 通过第 5.9 节讨论的延迟函数进行节流，特别是，每个副本将随着公证高度和认证高度之间的差距增加而增加调控器的值(这利用了局部调整延迟函数的概念，如第 5.12.2 节)。

  { State-speci c payload validation. As discussed in Section 5.7, the inputs in a pay-load must pass certain validity checks. In fact, these validity checks may depend to a certain degree on the state. A detail we skipped is that each block includes a round number, with the understanding that these validity checks should be made with respect to the certi ed state for that round number. A replica that needs to perform this validation will wait until the state for that round number has been certi ed, and then use the certi ed state for that round to perform the validation. This ensures that even with non-deterministic

computation, all replicas are performing the same validity tests (as otherwise, consensus could get stuck).

{国家特定有效载荷验证。正如第 5.7 节所讨论的，有效负载中的输入必须通过某些有效性检查。事实上，这些有效性检查可能在一定程度上取决于国家。我们跳过的一个细节是，每个块都包含一个整数，并且理解这些有效性检查应该针对该整数的已验证状态进行。需要执行此验证的副本将等待该轮数的状态被验证，然后使用该轮数的已验证状态执行验证。这确保了即使使用非确定性计算，所有的副本都执行相同的有效性测试(否则，一致性可能会卡住)。

6.2 Query calls vs update calls
6.2 查询调用 vs 更新调用

As we have described it so far, an ingress messages must pass through consensus so that they are processed in the same order by all replicas on a subnet. However, an important optimization is available to those ingress messages whose processing does not modify the replicated state of a subnet. These are called query calls | as opposed to other ingress messages, which are called update calls. Query calls are allowed to perform computations which read and possibly update the state of a canister, but any updates to the state of a canister are never committed to the replicated state. As such, a query call may be processed

正如我们到目前为止所描述的，入口消息必须通过共识，以便子网上的所有副本以相同的顺序处理它们。然而，一个重要的优化是可用于那些进入消息的处理没有修改子网的复制状态。这些被称为查询调用 | 与其他入口消息不同，后者被称为更新调用。允许查询调用执行读取并可能更新罐状态的计算，但是对罐状态的任何更新都不会提交给复制状态。因此，一个查询调用可以被处理

directly by a single replica without passing through consensus, which greatly reduces the latency for obtaining a response from a query call.
直接通过单个副本而不经过共识，这大大减少了从查询调用获得响应的延迟。

Note that a response to a query call is not recorded in the ingress history data structure. As such, we cannot directly use the per-round certi ed state mechanism to authenticate responses to query calls. However, a separate mechanism for authenticating such responses is provided: certi ed variables. As a part of the per-round certi ed state, each canister on a subnet is allocated a small number of bytes, which is the certi ed variable for that canister, whose value may be updated via update calls, and may be authenticated using the per-round certi ed state mechanism. Moreover, a canister may use its certi ed variable to store a root of a Merkle tree. In this way, a response to a query call to a canister may be authenticated so long the response is a leaf in the Merkle tree rooted at the certi ed variable for that canister.

请注意，对查询调用的响应不记录在入口历史数据结构中。因此，我们不能直接使用每轮验证状态机制来验证对查询调用的响应。然而，我们提供了一个独立的机制来验证这些响应: 验证变量。作为每轮认证状态的一部分，子网上的每个罐被分配少量字节，这是该罐的认证变量，其值可通过更新调用更新，并可使用每轮认证状态机制进行认证。此外，一个容器可以使用它的认证变量来存储 Merkle 树的根。通过这种方式，只要响应是默克尔树中的一片叶子，根植于该罐的已验证变量，那么对该罐的查询调用的响应就可以进行身份验证。

6.3 External user authentication
6.3 外部用户身份验证

One of the main di erences between an ingress message and a cross-subnet message is the mechanism used for authenticating these messages. We have already seen above (see Section 6.1) how threshold signatures are used to authenticate cross-subnet messages. The NNS registry (see Section 1.5) holds the public veri cation keys for the threshold signatures used to authenticate cross-subnet messages.
入口消息和跨子网消息的主要区别之一是用于认证这些消息的机制。我们已经在上面看到(见第 6.1 节)如何使用阈值签名来验证跨子网消息。NNS 注册表(参见第 1.5 节)持有用于认证跨子网消息的阈值签名的公共验证密钥。

There is no central registry for external users. Rather, an external user identi es himself to a canister using a user identi er, which is a hash of a public signature-veri cation key. The user holds a corresponding secret signing key, which is used to sign ingress messages. Such a signature, as well as the corresponding public key, is sent along with the ingress message. The IC automatically authenticates the signature and passes the user identi er to the appropriate canister. The canister may then authorize the requested operation, based on the user identi er and other parameters to the operation speci ed in the ingress message.

没有针对外部用户的中心注册表。相反，一个外部用户使用一个用户标识符(一个公共签名验证密钥的散列)将自己标识到一个容器中。用户持有相应的秘密签名密钥，用于签署进入消息。这样的签名，以及相应的公钥，与进入消息一起发送。IC 会自动认证签名，并将用户标识传递给相应的容器。然后，基于用户标识符和进入信息中指定的操作的其他参数，容器可以授权请求的操作。

First-time users generate a key pair and derive their user identi er from the public key during their rst interaction with the IC. Returning users are authenticated using the secret key that is stored by the user agent. A user may associate several key pairs with a single

user identity, using signature delegation. This is useful, as it allows a single user to access the IC from several devices using the same user identity.

第一次用户生成一个密钥对，并在第一次与 IC 交互时从公钥中获得用户标识。返回用户使用用户代理存储的密钥进行身份验证。一个用户可以使用签名委托将几个密钥对与一个用户身份关联起来。这很有用，因为它允许一个用户使用相同的用户标识从几个设备访问 IC。

7 Execution Layer
执行层

The execution environment processes one input at a time. This input is taken from one of the input queues, and is directed to one canister. Based on this input and the state of the canister, the execution environment updates the state of the canister, and additionally may add messages to output queues and update the ingress history (possibly with a response to an earlier ingress message).

执行环境一次处理一个输入。这个输入是从其中一个输入队列中获取的，然后被引导到一个容器中。执行环境根据这个输入和容器的状态更新容器的状态，另外还可以向输出队列添加消息并更新进入历史记录(可能是对早期进入消息的响应)。

In a given round, the execution environment will process several inputs. A scheduler determines which inputs are executed in a given round, and in which order. Without going into all the details of the scheduler, we highlight some of the goals:

在给定的一轮中，执行环境将处理几个输入。调度器决定在给定的一轮中执行哪些输入，以及执行的顺序。在不深入讨论调度器的所有细节的情况下，我们强调一些目标:

- it must be deterministic, i.e., only depend on the given data;
  它必须是确定性的，也就是说，仅仅依赖于给定的数据;

- it should distribute workloads fairly among canisters (but optimizing for throughput over latency).

它应该将工作负载公平地分配给各个容器(但要优化吞吐量，避免延迟)。

- the total amount of work done in each round, measured in terms of cycles (see Sec-tion 1.8), should be close to some pre-determined amount.

每一轮的总工作量，以周期计算(见第 1.8 节)，应接近预先订定的工作量。

Another task that the execution environment (together with the message router) must deal with are situations where a canister on one subnet is producing cross-subnet messages faster than they can be consumed by a canister on another subnet. For this, a self-regulating mechanism is implemented that throttles the producing canister.

执行环境(与消息路由器一起)必须处理的另一个任务是，一个子网上的一个容器产生跨子网消息的速度快于另一个子网上的容器消耗这些消息的速度。为此，一个自我调节机制被实现，以节流的生产罐。

There are many other resource management and bookkeeping tasks that are dealt with by the execution environment. However, all of these tasks must be dealt with determinis-tically.

还有许多其他资源管理和簿记任务是由执行环境处理的。然而，所有这些任务都必须确定性地处理。

7.1 Random tape

7.1 随机磁带

Each subnet has access to a distributed pseudorandom generator (PRG). As men-tioned in Section 3, pseudorandom bits are derived from a seed that itself is an (f + 1)-out-of-n BLS signature, called the Random Tape. There is a di erent Random Tape for each round of the consensus protocol. While this BLS signature is similar to that used for the Random Beacon used in consensus (see Section 5.5), the mechanics are somewhat di erent.

每个子网都可以访问一个分布式伪随机生成器(PRG)。正如第三部分中提到的，伪随机位是从一个种子中派生出来的，种子本身是一个(f + 1)-out-of-n BLS 签名，称为 Random Tape。共识协议的每一轮都有一个不同的随机磁带。虽然这个 BLS 签名与协商一致中使用的随机信标相似(见第 5.5 节)，但是机制有些不同。

In the consensus protocol, as soon as a block at height h is nalized, each honest replica will release its share of Random Tape for height h + 1. This has two implications:

在共识协议中，一旦处于高度 h 的块被整定，每个诚实的副本将释放其在高度 h + 1 上的 Random Tape 的份额。这有两个含义:

1. Before a block at height h is nalized by any honest replica, the Random Tape at height h + 1 is guaranteed to be unpredictable.

在任何诚实副本对高度 h 的块进行 nalized 之前，高度 h + 1 的 Random Tape 保证是不可预测的。

2. By the time block at height h + 1 is nalized by any honest replica, that replica will typically have all the shares it needs to construct the Random Tape at height h + 1.

通过高度 h + 1 的时间块被任何诚实的副本纳化，该副本通常拥有在高度 h + 1 构造 Random Tape 所需的所有共享。

To obtain pseudorandom bits, a subnet must make a request for these bits. Such a pseudorandom-bit request will be made as a \system call" from the execution layer in some round, say h. The system will then respond to that request later, when the Random Tape of height h + 1 is available. By property (1) above, it is guaranteed that the requested pseudorandom bits are unpredictable at the time the request is made. By property (2) above, the requested random bits will typically be available at the time the next block is nalized. In fact, in the current implementation, at the time a block of height h is nalized, the Consensus Layer (see Section 5) will deliver both (the payload of) the block of height h and the Random Tape of height h + 1 simultaneously to the message routing layer for processing.

为了获得伪随机位，子网必须对这些位发出请求。这样的一个伪随机位请求将作为一个系统调用从执行层进行一些回合，比如 h。之后，当随机磁带的高度 h + 1 可用时，系统会响应这个请求。通过上面的属性(1)，可以保证请求的伪随机位在请求发出时是不可预测的。上面的 By 属性(2)，请求的随机位通常在下一个块被 nalized 的时候是可用的。事实上，在当前的实现中，在对 height h 块进行处理时，Consensus 层(参见第 5 节)将同时将 height h 块和 height h + 1 的 Random Tape (有效负载)交付给消息路由层进行处理。

8 Chain-key cryptography II: chain-evolution technology
链式密钥加密技术 II: 链式进化技术

As mentioned in Section 1.6.2, chain-key cryptography includes a collection of technologies for robustly and securely maintaining a blockchain-based replicated state machine over time, which together form what is called chain-evolution technology. Each subnet operates

正如在第 1.6.2 节中提到的，链密钥加密包括一系列技术，用于随着时间的推移稳健和安全地维护基于区块链的复制状态机，这些技术共同形成了所谓的链进化技术。每个子网都在运行

in epochs of many rounds (typically on the order of a few hundreds of rounds). Chain-evolution technology implements many essential maintenance activities that are executed periodically with a cadence that is tied to epochs: garbage collection, fast forwarding, subnet membership changes, pro-active resharing of secrets, and protocol upgrades.
在许多回合的时代(通常在几百回合的顺序)。链进化技术实现了许多基本的维护活动，这些维护活动周期性地以与时代相关联的节奏执行: 垃圾收集、快速转发、子网成员变更、主动重新共享秘密和协议升级。

There are two essential ingredients to chain-evolution technology: summary blocks and catch-up packages (CUPs).
链式进化技术有两个基本要素: 汇总块和赶超包(CUPs)。

8.1 Summary blocks
8.1 摘要块

The rst block in each epoch is a summary block. A summary block contains special data that will be used to manage the shares of the various threshold signature schemes (see Section 3). There are two threshold schemes:
每个阶段的第一个块是摘要块。一个摘要块包含特殊的数据，这些数据将被用来管理各种阈值签名方案的共享(参见第 3 节)。有两种阈值方案:

- one (f + 1)-out-of-n scheme, for which a new signing key is generated every epoch;
一个(f + 1)-out-of-n 方案，每个时代生成一个新的签名密钥;

- one (n f)-out-of-n scheme, for which the signing key is reshared once every epoch.
一个(n f)-out-of-n 方案，对于该方案，签名键每个纪元重新刷新一次。

The low-threshold scheme is used for the random beacon and the random tape, while the high-threshold scheme is used to certify the replicated state of the subnet.
对随机信标和随机带采用低门限方案，对子网的复制状态采用高门限方案。

Recall that the DKG protocol (see Section 3.5) requires that for each signing key, we have a set of dealings, and that each replica can non-interactively obtain its share of the signing key from this set of dealings.
回想一下 DKG 协议(见第 3.5 节)要求对于每个签名密钥，我们有一组交易，每个副本可以非交互式地从这组交易中获得其签名密钥的份额。

Also recall that NNS maintains a registry that, among other things, determines the membership of a subnet (see Section 1.5). The registry (and hence the subnet membership) may change over time. Thus, subnets must agree on which registry version they use at various times for various purposes. This information is also stored in the summary block.
还要记住，NNS 维护一个注册表，该注册表决定子网的成员资格(参见第 1.5 节)。注册表(以及子网成员资格)可能会随着时间的推移而改变。因此，子网必须同意在不同时间为不同目的使用哪个注册表版本。这些信息也存储在汇总块中。

The summary block for epoch i contains the following data elds.
Epoch i 的摘要块包含以下数据区域。

- currentRegistryVersion. This registry version will determine the consensus committee used throughout epoch i | all tasks performed by the consensus layer (block making, notarization, nalization) will be performed by this committee.

当前注册版本。这个注册表版本将决定整个时代使用的协商一致委员会 | 协商一致层执行的所有任务(块制作、公证、化)将由这个委员会执行。

- nextRegistryVersion. In each round of consensus, a block maker will include in its proposal the latest registry version it knows about (which must be no earlier than the block the proposed block extends). This ensures that the value nextRegistryVersion in the summary block of epoch i is fairly up to date.

nextRegistryVersion.在每一轮共识中，块制造商将在其提案中包括它所知道的最新注册表版本(必须不早于提议块扩展的块)。这样可以确保 epoch i 摘要块中的 nextRegistryVersion 值是最新的。

The value of currentRegistryVersion in epoch i is set to the value of nextRegistryVer-sion in epoch i 1.
currentRegistryVersion 在 epoch i 中的值设置为 nextRegistryVer-sion 在 epoch i 1 中的值。

- currentDealingSets. These are the dealing sets that determine the threshold signing keys that will be used to sign messages in epoch i.

currentDealingSets 当前交易集。这些处理集决定了在第一纪元中用于对消息进行签名的阈值签名键。

As we will see, the threshold signing committee for epoch i (i.e., the replicas that hold the corresponding threshold signing key shares) is the consensus committee for epoch i 1 .
正如我们将看到的，epoch i 的阈值签名委员会(即，持有相应的阈值签名密钥份额的副本)是 epoch i 1 的协商一致委员会。

- nextDealingSets. This is where dealings that are collected during epoch i 1 are gathered and stored.[7] The value of currentDealingSets in epoch i will be set to the value of nextDealingSets in epoch i 1 (which itself consists of dealings collected in epoch i 2).

· 下一个交易集。这是收集和存储在历元 i 1 期间收集的交易的地方。7. 历元 i 中的 currentDealingSets 的值将被设置为历元 i 1 中的 nextDealingSets 的值(它本身包括在历元 i 2 期间收集的交易)。

- collectDealingParams. This describes the parameters that de ne the dealing sets to be collected during epoch i. During epoch i, block makers will include dealings in their proposed blocks that are validated relative to these parameters.

collectDealingParams.这描述了在第一纪期间收集的交易集的参数。在第一阶段，区块制造商将包括交易在他们提出的区块，被验证相对于这些参数。

The receiving committee for these dealings is based on the nextRegistryVersion value of the summary block of epoch i.
这些交易的接收委员会基于 epoch i 摘要块的 nextRegistryVersion 值。

For the low-threshold scheme, the dealing committee is the consensus committee for epoch i.
对于低门槛方案，交易委员会是时代 i 的共识委员会。

For the high-threshold scheme, the shares to be reshared are based on the value of nextDealingSets of epoch i. Therefore, the dealing committee is the receiving committee for epoch i 1, which is also the consensus committee for epoch i .
对于高阈值方案，需要重新配置的股票是基于时代 i 的下一个交易集的值，因此，交易委员会是时代 i 1 的接收委员会，也是时代 i 的协商委员会。

Also observe that the threshold signing committee for epoch i is the receiving committee in epoch i 2, which is the consensus committee for epoch i 1 .
还可以看到，第 i 纪的门槛签字委员会是第 i 2 纪的接受委员会，也就是第 i 1 纪的共识委员会。

Consensus in epoch i relies on the values currentRegistryVersion and currentDealingSets in epoch i | in particular, the makeup of the consensus committee itself is based on curren-tRegistryVersion and the random beacon used in consensus is based on currentDealingSets. Moreover, just like any other block, there could be more than one summary block notarized at the beginning of epoch i, and that ambiguity needs to be resolved by consensus in epoch i. This seeming circularity is resolved by insisting that a summary block at the beginning of epoch i 1 has been nalized before epoch i starts, since the relevant values in the newer summary block are copied directly from that older summary block. This is actually an implicit synchrony assumption, but it is quite an academic assumption. Indeed, because of the \consensus throttling" discussed in Section 5.12.2 to ensure liveness, and because of the length of an epoch is quite large, this can essentially never happen in practice: long before consensus could reach the end of epoch i 1 without nalizing a summary block for epoch i 1, the notarization delay function would grow to be astronomically large, and so the partial synchrony assumption needed for nalization will be satis ed (essentially) with certainty (for all practical purposes).[8]

特别是，协商一致委员会本身的组成是以协商一致版本为基础的，协商一致中使用的随机信标是以协商一致版本为基础的。此外，就像任何其他块，可以有一个以上的摘要块公证在第一纪的开始，歧义需要解决协商一致在第一纪。通过坚持在 epoch i 开始之前已经对 epoch i 1

开始处的摘要块进行了初始化，可以解决这种表面上的循环，因为新摘要块中的相关值直接从旧摘要块复制。这实际上是一个隐式的同步假设，但它是一个相当理论化的假设。事实上，由于第 5.12.2 节讨论的“共识节流”以确保生动性，而且由于一个时代的长度相当大，这在实践中基本上永远不可能发生: 早在共识可以达到第 i 1 时代的末尾而不对第 i 1 时代进行总结块之前，公证延迟函数就会增长到天文数字般的大，因此，对于所有实际目的而言，实现所需的部分同步假设基本上将得到确定的满足。8

8.2 CUPs
8.2 杯

Before describing a CUP, we rst point out one detail of random beacon: the random beacon for each round depends on the random beacon for the previous round. This is not an essential feature, but it impacts the design of the CUP.
在描述 CUP 之前，我们首先指出随机信标的一个细节: 每轮的随机信标取决于前一轮的随机信标。这不是一个重要的特征，但是它影响了 CUP 的设计。

[7]A detail we have omitted is that if we fail to collect all the required dealings in epoch i 1, then as a fallback, the value of nextDealingSets in epoch i will e ectively be set to the value of currentDealingSets in epoch i. If this happens, then the protocol will make use of dealing committees and threshold signing committees from further in the past, as appropriate.
我们忽略的一个细节是，如果我们不能收集所有在第 i 1 时代需要的交易，那么作为一个备用方案，第 i 时代的 nextDealingSets 的值将分别设置为第 i 时代的 currentDealingSets 的值。如果发生这种情况，那么协议将酌情利用过去的交易委员会和阈值签署委员会。

[8]Also note that dealings that are collected in epoch i depend on data in the summary block for epoch i, in particular, the values of nextDealingSets and nextRegistryVersion . As such, these dealings should not be generated and cannot be validated until a summary block for epoch i has been nalized.
还要注意，在 epoch i 中收集的事务依赖于 epoch i 摘要块中的数据，特别是 nextDealingSets 和 nextRegistryVersion 的值。因此，这些事务不应该被生成，并且在纪元 i 的汇总块被定义之前不能被验证。

40

A CUP is a special message (not on the blockchain) that has (mostly) everything a replica needs to begin working in a given epoch, without knowing anything about previous epochs. It consists of the following data elds:

CUP 是一个特殊的消息(不在区块链上)，它(大多数情况下)具有副本在给定时代开始工作所需的一切，而不需要知道任何关于以前时代的信息。它由以下数据组成:

- The root of a Merkle hash tree for the entire replicated state (as opposed to the partial, per-round certi ed state as in Section 6.1).

Merkle 散列树的根用于整个复制状态(与第 6.1 节中的部分、每轮验证状态相反)。

- The summary block for the epoch.

纪元的摘要块。

- The random beacon for the rst round of the epoch.

纪元第一轮的随机信标。

- A signature on the above elds under the (n f)-out-of-n threshold signing key for the subnet.

•子网的(n f)-out-of-n 阈值签名键下的上述电平上的签名。

To generate a CUP for a given epoch, a replica must wait until the summary block for that epoch is nalized and the corresponding per-round state is certi ed. As already mentioned, the entire replicated state must be hashed as a Merkle tree | even though a number of techniques are used to accelerate this process, this is still quite expensive, which is why it is only done once per epoch. Since a CUP contains only the root of this Merkle tree, a special state sync subprotocol is used that allows a replica to pull any state that it needs from its peers | again, a number of techniques are used to accelerate this process, but it is still quite expensive. Since we are using a high-threshold signature for a CUP, we can be sure that there is only one valid CUP in any epoch, and moreover, there will be many peers from which the state may be pulled.

为了为给定的纪元生成 CUP，副本必须等待，直到该纪元的汇总块被初始化，并且相应的每轮状态被确认。正如前面提到的，整个复制状态必须作为 Merkle 树 | 进行散列，即使使用了许多技术来加速这个过程，这仍然是相当昂贵的，这就是为什么它每个纪元只执行一次的原因。由于 CUP 只包含这个 Merkle 树的根，因此使用了一个特殊的状态同步子协议，允许副本从它的同伴那里提取它需要的任何状态。同样，许多技术被用来加速这个过程，但是它仍然相当昂贵。由于我们正在为 CUP 使用一个高阈值签名，因此我们可以确定在任何时代只有一个有效的 CUP，而且可以从许多对等点中提取状态。

8.3 Implementing chain-evolution technology

8.3 实施连锁进化技术

Garbage collection: Because of the information contained in a CUP for a given epoch, it is safe for each replica to purge all inputs that have been processed, and all consensus-level protocol messages needed to order those inputs, prior to that epoch.

垃圾收集: 由于给定纪元的 CUP 中包含的信息，因此在该纪元之前，每个副本可以安全地清除已经处理的所有输入以及排序这些输入所需的所有共识级协议消息。

Fast forwarding: If a replica in a subnet falls very far behind its peers (because it is down or disconnected from the network for a long time), or a new replica is added to a subnet, it

can be fast forwarded to the beginning of the most recent epoch, without having to run the consensus protocol and process all of the inputs up to that point. Such a replica may do so by obtaining the most recent CUP. Using the summary block and random beacon contained in the CUP, along with protocol messages from other replicas (which have not yet been purged), this replica may run the consensus protocol forward from the beginning of the corresponding epoch. The replica will also use the state sync subprotocol to obtain the replicated state corresponding to the beginning of the epoch, so that it may also process the inputs generated by consensus.

快速转发：如果子网中的一个副本远远落后于它的同伴(因为它长时间处于停机状态或与网络断开连接)，或者一个新的副本被添加到一个子网中，那么它就可以快速转发到最近一个纪元的开始，而不必运行共识协议并处理到那个时候为止的所有输入。这样的副本可以通过获得最新的 CUP 来实现。使用 CUP 中包含的汇总块和随机信标，以及来自其他副本(尚未清除)的协议消息，该副本可以从相应时代的开始向前运行共识协议。副本还将使用状态同步子协议来获得与纪元开始相对应的复制状态，以便它也可以处理由共识生成的输入。

Figure 5 illustrates fast forwarding. Here, we assume that a replica that needs to catch up has a CUP at the beginning of an epoch, which starts (say) at height 101. The CUP contains the root of the Merkle tree for the replicated state at height 101, the summary block at height 101 (shown in green), and the random beacon at height 101. This replica will use the state sync subprotocol to obtain from its peers the full replicated state at height 101, using the root of the Merkle tree in the CUP to validate this state.

图 5 说明了快进。在这里，我们假设一个需要赶上的副本在一个纪元的开始处有一个 CUP，它从高度 101 开始。CUP 包含 Merkle 树的根，用于在高度 101 处的复制状态，在高度 101 处的汇总块(以绿色显示)，以及在高度 101 处的随机信标。这个副本将使用状态同步子协议从它的对等点获得高度为 101 的完整复制状态，使用 CUP 中 Merkle 树的根来验证这个状态。

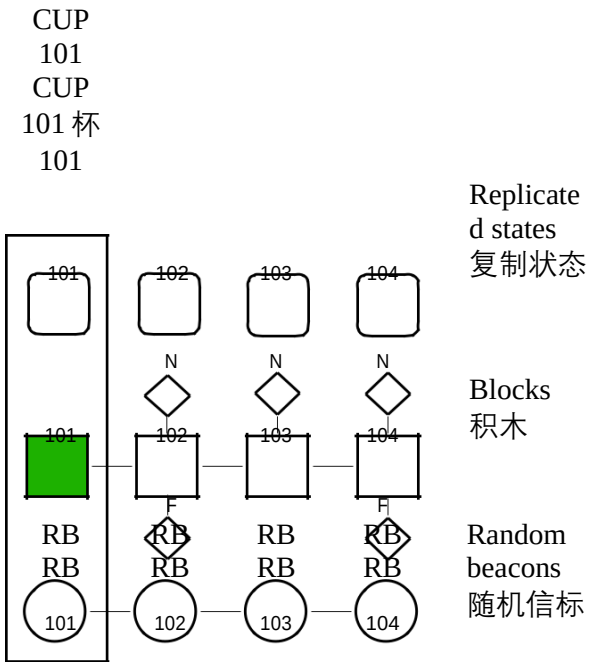

Figure 5: Fast forwarding
图 5: 快进

Having obtained this state, the replica can then participate in the protocol, obtaining from its peers blocks (and other messages associated with consensus) at heights 102, 103, and so on, and updating its copy of the replicated state. If its peers have already nalized blocks at greater heights, this replica will process those nalized blocks as quickly as it can obtain them (and their notarizations and nalizations) from its peers (and as quickly as the execution layer will allow).
获得这个状态之后，副本就可以加入协议，从高度为 102、103 等的对等块(以及与共识相关的其他消息)获取信息，并更新其复制状态的副本。如果它的对等方已经在更高的高度上对数据块进行了数据化，那么这个副本将尽可能快地处理这些数据块(以及它们的公证和数据化)。

Subnet membership changes: We have already discussed how summary blocks are used to encode which version of the registry is in force in a given epoch, and how that is used to determine the subnet membership, and more speci cally, the membership committees for various tasks. Note that even after a replica is removed from a subnet, it should (if possible) participate in its assigned committee duties for one additional epoch .
子网成员资格的变化: 我们已经讨论了如何使用汇总块来编码注册中心在给定时代的生效版本，以及如何使用汇总块来确定子网成员资格，更具体地说，确定各种任务的成员委员会。请注意，即使一个副本从子网中删除，它也应该(如果可能的话)参与一个额外时代的委员会任务。

Pro-active resharing of secrets: We have already discussed how summary blocks are used to generate and reshare signing keys. If necessary, the required summary block may be obtained from a CUP.

Pro-active resharing of secrets: 我们已经讨论了如何使用汇总块来生成和重新共享签名密钥。如果有必要，所需的摘要块可以从 CUP 获得。

Protocol upgrades: CUPs are also used to implement protocol upgrades. Protocol upgrades are initiated by the NNS (see Section 1.5). The basic idea, without going into all the details, is this:

Protocol 升级: CUPs 也用于实现协议升级。协议升级是由 NNS 发起的(参见 1.5 节)。基本概念，不涉及所有细节，是这样的:

- when it is time to install a new version of the protocol, the summary block at the beginning of an epoch will indicate this;

  当需要安装新版本的协议时，新纪元开头的汇总块会指出这一点;

- the replicas running the old version of the protocol will continue running consensus long enough to nalize the summary block and to create a corresponding CUP; however, they will create only empty blocks and not pass along any pay-loads to message routing and execution;

  运行旧版本协议的副本将继续运行协商一致足够长的时间来实现摘要块并创建相应的 CUP; 然而，它们将只创建空块，并且不会将任何有效负载传递给消息路由和执行;

42

- the new version of the protocol will be installed, and the replicas running the new version of the protocol will resume running the full protocol from the above CUP.

将安装新版本的协议，运行新版本的协议的副本将恢复运行上述 CUP 的完整协议。

References
参考文献

[AMN+20] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin. Sync HotStu : Simple and Practical Synchronous State Machine Replication. In 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020, pages 106{118. IEEE, 2020.

[AMN + 20] i. Abraham，d. Malkhi，k. Nayak，l. Ren，and m. Yin.Sync HotStu: 简单实用的同步状态机复制。2020 年 IEEE 安全与隐私论坛，SP 2020，旧金山，加利福尼亚州，美国，2020 年 5 月 18-21 日，第 106 页{118}。IEEE，2020.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Veri ably Encrypted Signatures from Bilinear Maps. In E. Biham, editor, Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings, volume 2656 of Lecture Notes in Computer Science, pages 416{ 432. Springer, 2003.

D. Boneh，c. Gentry，b. Lynn，and h. Shacham.来自双线性地图的聚合和真正加密的签名。2003 年 5 月 4 日至 8 日在波兰华沙召开的密码技术理论与应用国际会议，《计算机科学讲义》第 2656 卷，第 416 页。斯普林格，2003。

[BKM18] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus,
[BKM18] E. Buchman，j. Kwon，and z. Milosevic。关于 BFT 共识的最新八卦，
2018. arXiv:1807.04938, http://arxiv.org/abs/1807.04938.
2018. arXiv: 1807.04938，http://arXiv.org/abs/1807.04938.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing.
[BLS01] D. Boneh b. Lynn 和 h. Shacham. 油井配对的简短签名。
In C. Boyd, editor, Advances in Cryptology - ASIACRYPT 2001, 7th Interna-
在 C.Boyd，编辑，密码学进展 -ASIACRYPT 2001，第 7 届国际
tional Conference on the Theory and Application of Cryptology and Information
密码学和信息学理论与应用传统会议
Security, Gold Coast, Australia, December 9-13, 2001, Proceedings, volume 2248
安全，黄金海岸，澳大利亚，2001 年 12 月 9 日至 13 日，会议记录，卷 2248
of Lecture Notes in Computer Science, pages 514{532. Springer, 2001.
《计算机科学讲义》第 514 页{532. Springer，2001。

[But13] V. Buterin. Ethereum whitepaper, 2013. https://ethereum.org/en/
[但 13] 布特林。以太网白皮书，2013。Https://ethereum.org/en/
whitepaper/.
白皮书/。

[CDH+21] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup, and D. Williams. Internet Computer Consensus. Cryptology ePrint Archive, Report 2021/632, 2021. https://ia.cr/2021/632.
[CDH + 21] j. Camenisch， m. Drijvers， t. Hanke， y.a.Pignolet， v. Shoup 和 d. Williams。Internet Computer Consensus 互联网计算机共识。Cryptology ePrint Archive，Report 2021/632,2021.Https://ia.cr/2021/632.

[CDS94] R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of Partial Knowledge and Simpli ed Design of Witness Hiding Protocols. In Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings, volume 839 of Lecture Notes in Computer Science, pages 174{187. Springer, 1994.
[CDS94] r. Cramer，i. Damgard，and b. Schoenmakers.证明部分知识和证人隐藏协议的 Simpli 设计。密码学进展 -CRYPTO'94，第 14 届国际密码讨论年会，Santa Bar-bara， California， USA， August 21-25,1994， Proceedings， volume 839 of Lecture Notes In Computer Science， pages 174{187。Springer 出版社，1994 年。

[CL99] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In M. I. Seltzer and P. J. Leach, editors, Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999, pages 173{186. USENIX Association, 1999.
卡斯特罗先生和李斯科夫先生。实用的拜占庭将军问题容错。在 M.I.Seltzer 和 P.J.Leach，编辑，关于操作系统设计和实现(OSDI)的 USENIX 第三次研讨会会议记录，美国路易斯安那州新奥尔良，1999 年 2 月 22-25 日，第 173 页{186}。USENIX 协会，1999。

[CWA+09] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In J. Rexford and
[CWA + 09] a. Clement，e. l. Wong，l. Alvisi，m. Dahlin，and m. Marchetti.使拜占庭容错系统容忍拜占庭错误。在 J.Rexford 和

E. G. Sirer, editors, Proceedings of the 6th USENIX Symposium on Networked Sirer，editor，Proceedings of the 6th USENIX Symposium on Networked
Systems Design and Implementation, NSDI 2009, April 22-24, 2009, Boston, 系统设计与实现，NSDI 2009,2009 年 4 月 22-24，波士顿,
MA, USA, pages 153{168. USENIX Association, 2009. http://www.usenix. 美国马萨诸塞州，第 153 页{168。 USENIX 协会，2009。 http://www.USENIX。
org/events/nsdi09/tech/full_papers/clement/clement.pdf. Org/events/nsdi09/tech/full _ papers/clement/clement.pdf.

[Des87] Y. Desmedt. Society and Group Oriented Cryptography: A New Concept. In
[Des87] y。 Desmedt。面向社会和群体的密码学: 一个新概念
C. Pomerance, editor, Advances in Cryptology - CRYPTO '87, A Conference 编辑，密码学进展 -CRYPTO'87，一次会议
on the Theory and Applications of Cryptographic Techniques, Santa Barbara, 加密技术的理论和应用,
California, USA, August 16-20, 1987, Proceedings, volume 293 of Lecture Notes 美国加利福尼亚州，1987 年 8 月 16 日至 20 日，会议录，讲义第 293 卷
in Computer Science, pages 120{127. Springer, 1987.
计算机科学，第 120 页{127. Springer，1987。

[DLS88] C. Dwork, N. A. Lynch, and L. J. Stockmeyer. Consensus in the presence of
[DLS88] c. d. 工作，N.a. 林奇，和 l. j. 斯托克迈耶
partial synchrony. J. ACM, 35(2):288{323, 1988.
部分同步.j. ACM，35(2) : 288{323,1988。

[Fis83] M. J. Fischer. The Consensus Problem in Unreliable Distributed Systems (A
不可靠分布式系统中的共识问题
Brief Survey). In Fundamentals of Computation Theory, Proceedings of the 1983 简要概述)。《计算理论基础》，1983 年会议录
International FCT-Conference, Borgholm, Sweden, August 21-27, 1983, volume 国际 fct 会议，博里霍尔姆，瑞典，1983 年 8 月 21 日至 27 日，卷
158 of Lecture Notes in Computer Science, pages 127{140. Springer, 1983.
计算机科学讲义 158 页 127{140. Springer，1983。

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identi ca-
[FS86] a. Fiat 和 a. Shamir 如何证明自己: 识别身份的实用方法
tion and Signature Problems. In Advances in Cryptology - CRYPTO '86, Santa 《密码学进展》(In Advances In Cryptology)-86 年《加密》(CRYPTO'86)，圣诞老人
Barbara, California, USA, 1986, Proceedings, volume 263 of Lecture Notes in 芭芭拉，加利福尼亚州，美国，1986 年，会议录，第 263 卷的演讲笔记
Computer Science, pages 186{194. Springer, 1986.
计算机科学，第 186 页{194. Springer，1986。

[GHM+17] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. https://eprint.iacr.org/2017/454.
[GHM + 17] y. Gilad，r. Hemo，s. Micali，g. Vlachos，and n. Zeldovich.Algorand: 缩放加密货币的拜占庭协议。Cryptology ePrint Archive，Report 2017/454,2017.Https://eprint.iacr.org/2017/454.

[Gro21] J. Groth. Non-interactive distributed key generation and key resharing. Cryp-
[Gro21] j. Groth. 非交互式分布式密钥生成和密钥重新共享. Cryp -
tology ePrint Archive, Report 2021/339, 2021. https://ia.cr/2021/339.
Tology ePrint Archive，Report 2021. https://ia.cr/2021/339.

[JMV01] D. Johnson, A. Menezes, and S. A. Vanstone. The Elliptic Curve Digital Sig-
D. Johnson，a. Menezes 和 S.a. Vanstone。椭圆曲线数字签名
nature Algorithm (ECDSA). Int. J. Inf. Sec., 1(1):36{63, 2001.
自然算法(ECDSA) . Inf sec.，1(1) : 36{63,2001。

[Mer87] R. C. Merkle. A Digital Signature Based on a Conventional Encryption Func-
[Mer87] R.c。 Merkle。基于常规加密 Func-的数字签名
tion. In Advances in Cryptology - CRYPTO ’87, A Conference on the Theory
密码学进展-87 年密码学理论研讨会
and Applications of Cryptographic Techniques, Santa Barbara, California, USA,
和密码技术的应用，美国加利福尼亚州圣巴巴拉市，
August 16-20, 1987, Proceedings, volume 293 of Lecture Notes in Computer Sci-
1987 年 8 月 16-20 日，《计算机科学讲义》第 293 卷
ence, pages 369{378. Springer, 1987.
第 369 页{378}，Springer，1987。

[MXC+16] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The Honey Badger of BFT Protocols. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. My-ers, and S. Halevi, editors, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, pages 31{42. ACM, 2016.
[MXC + 16] a. Miller，y. Xia，k. Croman，e. Shi，and d. Song.BFT 协议的蜜獾。2016 年 ACM SIGSAC 计算机与通信安全会议论文集，奥地利维也纳，2016 年 10 月 24 日至 28 日，第 31 页{42}。ACM，2016.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. https:
比特币: 一个点对点的电子现金系统，2008:
//bitcoin.org/bitcoin.pdf.
//bitcoin.org/bitcoin.pdf.

[PS18] R. Pass and E. Shi. Thunderella: Blockchains with Optimistic Instant Con-
[PS18] r. Pass 和 e. Shi. thunderrella: 基于乐观即时控制的区块链
rmation. In J. B. Nielsen and V. Rijmen, editors, Advances in Cryptology -
在 J.b. 尼尔森和 v. Rijmen，编辑，密码学的进展
EUROCRYPT 2018 - 37th Annual International Conference on the Theory and
EUROCRYPT 2018-37 年度理论与技术国际会议
Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3,
加密技术的应用，以色列特拉维夫，4 月 29 日至 5 月 3 日，
2018 Proceedings, Part II, volume 10821 of Lecture Notes in Computer Science,
《2018 年计算机科学讲义》第 10821 卷第二部分，
pages 3{33. Springer, 2018.
第 3 页{33. Springer，2018。

[PSS17] R. Pass, L. Seeman, and A. Shelat. Analysis of the Blockchain Protocol in
[PSS17] r. Pass，l. Seeman，and a. Shelat. 块链协议的分析
Asynchronous Networks. In Advances in Cryptology - EUROCRYPT 2017 -
异步网络。在密码学的进展-EUROCRYPT 2017-
36th Annual International Conference on the Theory and Applications of Cryp-
第 36 届 Cryp 理论与应用国际年会
tographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part
2017 年 4 月 30 日至 5 月 4 日，法国巴黎
II, volume 10211 of Lecture Notes in Computer Science, pages 643{673, 2017.
计算机科学讲义第 10211 卷，第 643 页{673,2017。

[Sch90] F. B. Schneider. Implementing Fault-Tolerant
Services Using the State Machine Approach: A Tutorial. ACM
Comput. Surv., 22(4):299{319, 1990.
[Sch90] f. b. Schneider. 使用状态机方法实现容错服务: 一个教程.
ACM 计算. surv.，22(4) : 299{319,1990。

45