The Libra BlockChain(中英混合版)

> The Libra BlockChain(中文版)

Abstract

The Libra Blockchain is a decentralized, programmable database designed to support a low-volatility cryptocurrency that will have the ability to serve as an efficient medium of exchange for billions of people around the world. We present a proposal for the Libra protocol, which implements the Libra Blockchain and aims to create a financial infrastructure that can foster innovation, lower barriers to entry, and improve access to financial services. To validate the design of the Libra protocol, we have built an open-source prototype implementation — Libra Core — in anticipation of a global collaborative effort to advance this new ecosystem.
Libra区块链是一个分散的、可编程的数据库，旨在支持低波动性的密码货币，该货币将有能力作为全球数十亿人的高效交换媒介。我们为Libra协议提出了一个提案，该协议实现了Libra区块链，旨在创建一个金融基础设施，可以促进创新、降低进入壁垒和改善金融服务的获取。为了验证Libra协议的设计，我们构建了一个开源原型实现——Li ra核心——期待全球合作努力推进这个新的生态系统。

The Libra protocol allows a set of replicas — referred to as validators — from different authorities to jointly maintain a database of programmable resources. These resources are owned by different user accounts authenticated by public key cryptography and adhere to custom rules specified by the developers of these resources. Validators process transactions and interact with each other to reach consensus on the state of the database. Transactions are based on predefined and, in future versions, user-defined smart contracts in a new programming language called Move.
Libra 协议允许来自不同权威机构作为区块链复制节点-被称为验证器-共同维护一个可编程资源的数据库。这些资源由不同的用户账户拥有，通过公钥密码学认证，并遵守这些资源的开发人员指定的自定义规则。验证器处理交易并相互交互，以就数据库的状态达成共识。交易基于预定义，在未来的版本中，用户定义的智能合约使用一种新的编程语言，称为Move。

We use Move to define the core mechanisms of the blockchain, such as the currency and validator membership. These core mechanisms enable the creation of a unique governance mechanism that builds on the stability and reputation of existing institutions in the early days but transitions to a fully open system over time.

我们使用 Move 来定义区块链的核心机制，如货币和验证者成员资格，这些核心机制能够创建一种独特的治理机制，这种机制在早期建立在现有机构的稳定性和声誉的基础上，但随着时间的推移过渡到完全开放的系统。

1. Introduction

The spread of the internet and resulting digitization of products and services have increased efficiency, lowered barriers to entry, and reduced costs across most industries. This connectivity has driven economic empowerment by enabling more people to access the financial ecosystem. Despite this progress, access to financial services is still limited for those who need it most — impacted by cost, reliability, and the ability to seamlessly send money.
互联网的普及以及由此产生的产品和服务数字化提高了效率，降低了进入壁垒，并降低了大多数行业的成本。这种连接通过使更多人能够进入金融生态系统来推动经济赋权。尽管取得了这一进展，但对那些最需要金融服务的人来说，获得金融服务的机会仍然有限——受到成本、可靠性和无缝发送资金的能力的影响。

This paper presents a proposal for the Libra protocol, which supports the newly formed Libra ecosys- tem that seeks to address these challenges, expand access to capital, and serve as a platform for innovative financial services. This ecosystem will offer a new global currency — the Libra coin —which will be fully backed with a basket of bank deposits and treasuries from high-quality central banks. All of these currencies experience relatively low inflation, and thus the coin mechanically inherits this property as well as the advantages of a geographically diversified portfolio of assets. The Libra protocol must scale to support the transaction volume necessary for this currency to grow into a global financial infrastructure and provide the flexibility to implement the economic and governance policies that support its operations. The Libra protocol is designed from the ground up to holisti- cally address these requirements and build on the learnings from existing projects and research — a combination of novel approaches and well-understood techniques.
本文提出了Libra协议的建议，该协议支持新成立的Libra生态系统，旨在解决这些挑战，扩大资本准入，并作为创新金融服务的平台。 这个生态系统将提供一种新的全球货币——Libra Coin——它将由一篮子银行存款和高质量中央银行的国债提供充分支持。 所有这些货币都经历了相对较低的通货膨胀，因此Libra自然地继承了这种财产以及地理上多样化的资产组合的优势。 Libra协议必须扩大规模，以支持这种货币成长为全球金融基础设施所需的交易量，并提供实施支持其业务的经济和治理政策的灵活性。 Libra协议是从根本上设计的，从整体上解决这些需求，并建立在现有项目和研究的学习基础上——这是一种新颖的方法和广为人知的技术的结合。

A key prerequisite for healthy competition and innovation in financial services is the ability to rely on common infrastructure for processing transactions, maintaining accounts, and ensuring interoper- ability across services and organizations. By lowering barriers to entry and switching costs, the Libra protocol will enable startups and incumbents to compete on a level playing field, and experiment with new types of business models and financial applications. Blockchain technology lends itself well to address these issues because it can be used to ensure that no single entity has control over the ecosystem or can unilaterally shape its evolution to its advantage [1].
金融服务良性竞争和创新的一个关键先决条件是能够依赖共同的基础设施来处理交易、维护账户，并确保跨服务和组织的互操作性。 通过降低进入和交换成本的障碍， Libra 协议将使初创公司和在职者能够在一个公平的竞争环境中竞争，并试验新类型的商业模式和金融应用。 区块链技术很好地利用了自己来解决这些问题，因为它可以用来确保没有任何一个实体对生态系统拥有控制权，或者可以单方面塑造它的进化以达到其优势。[1]。

The Libra Blockchain will be decentralized, consisting of a collection of validators that work together to process transactions and maintain the state of the blockchain. These validators also form the membership of the Libra Association, which will provide a framework for the governance of the network and the reserve that backs the coin. Initially, the association (and validators) will consist of a geographically distributed and diverse set of Founding Members. These members are organizations chosen according to objective participation criteria, including that they have a stake in bootstrapping the Libra ecosystem and investing resources toward its success. Over time, membership eligibility will shift to become completely open and based only on the member’s holdings of Libra. The association has published reports outlining its vision [2], its proposed structure [3], the coin’s economics [4], and the roadmap for the shift toward a permissionless system [5].
Libra区块链将是去中心化的，由一个验证器的集合组成，这些验证器一起工作来处理交易和维护区块链的状态。 这些验证者也构成了LIbra协会的成员，它将为网络的治理和支持硬币的储备提供一个框架。 最初，该协会(和验证者)将由地理分布和不同的创始成员组成。 这些成员是根据客观的参与标准选择的组织，包括他们在引导Libra生态系统和为其成功投资资源方面有利害关系。 随着时间的推移，会员资格将变得完全开放，并且只基于会员对天秤座的持有量。 该协会发表了报告，概述了其愿景[2]、提议的结构[3]、硬币的经济学[4]，以及向无限制系统转变的路线图[5]。

This paper is the first step toward building a technical infrastructure to support the Libra ecosystem. We are publishing this early report to seek feedback from the community on the initial design, the plans for evolving the system, and the currently unresolved research challenges discussed in the proposal. Thus, the association has established an open-source community [6] for the discussion and development of the project.

本文是构建支持Libra生态系统的技术基础设施的第一步。我们正在发布这份早期报告，寻求社区对初步设计、系统发展计划以及提案中讨论的目前尚未解决的研究挑战的反馈。因此，该协会建立了一个开源社区[6]，用于项目的讨论和开发。

The Libra protocol. The Libra Blockchain is a cryptographically authenticated database [7, 8, 9] maintained using the Libra protocol. The database stores a ledger of programmable resources, such as Libra coins. A resource adheres to custom rules specified by its declaring module, which is also stored in the database. A resource is owned by an account that is authenticated using public key cryptography. An account could represent direct end users of the system as well as entities, such as custodial wallets, that act on behalf of their users. An account’ s owner can sign transactions that operate on the resources held within the account.
天秤座协议. Libra Blockchain 是使用 Libra 协议维护的密码认证数据库[7，8，9]。 数据库存储可编程资源的账本，如Libra Coins。 资源遵守其声明模块指定的自定义规则，该模块也存储在数据库中。 资源由使用公钥密码学认证的帐户拥有。 账户可以代表系统的直接最终用户以及代表用户行事的实体，如保管钱包。 账户的所有者可以签署在账户中持有的资源上操作的交易。

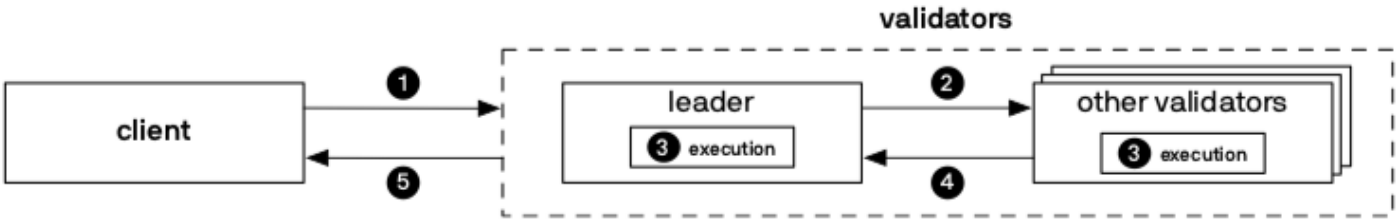

Figure 1: Overview of the Libra protocol.

Figure 1 shows the two types of entities that interact using the Libra protocol: (1) validators, which maintain the database and (2) clients, which perform queries on the database and submit transactions to modify it.
图1显示了使用 Libra 协议进行交互的两类实体：（1）验证器，维护数据库和（2）客户端，客户端对数据库执行查询，并提交事务以修改。

Validators maintain the database and process transactions submitted by clients for inclusion in the database (1). The validators use a distributed consensus protocol to agree on an ever-growing list of transactions that have been committed to the database as well as the results of executing those transactions. This consensus protocol must be reliable even in the presence of malicious or erroneous behavior by a minority of validators. Validators take turns driving the process of accepting transac- tions. When a validator acts as a leader, it proposes transactions, both those directly submitted to it by clients and those indirectly submitted through other validators, to the other validators (2). All validators execute the transactions (3) and form an

authenticated data structure that contains the new ledger history. The validators vote on the authenticator for this data structure as part of the consensus protocol (4). As part of committing a transaction Ti at version i, the consensus protocol outputs a signature on the full state of the database at version i — including its entire history — to authenticate responses to queries from clients (5).
验证器维护数据库，并处理客户端提交的事务，以便纳入数据库(1)。 验证者使用分布式共识协议来商定一个不断增加的已承诺到数据库的教育列表以及执行这些交易的结果。 即使在少数验证者恶意或错误行为的情况下，这种共识协议也必须是可靠的。 验证者轮流推动接受交易的过程。 当验证器作为领导者时，它向其他验证器提议交易，包括客户直接提交给它的交易和通过其他验证器间接提交的交易(2)。所有验证器执行事务(3)，并形成一个包含新分类账历史的经过认证的数据结构。 作为共识协议(4)的一部分，验证者对该数据结构的认证者进行投票。 作为在版本 i 提交事务 Ti 的一部分，共识协议在版本 i 输出数据库的全部状态签名(包括其全部历史记录)，以验证对客户端查询的响应(5)。

Clients can issue queries to a validator to read data from the database. Since the database is authen- ticated, clients can be assured of the accuracy of the response to their query. As part of the response to a read query, a validator returns a signed authenticator for the latest version i of the database known to the validator.
客户端可以向验证器发出查询，以便从数据库读取数据。由于数据库是经过验证的，客户端可以保证对其查询的响应的准确性。作为对读取查询的响应的一部分，验证器返回验证器所知数据库的最新版本 i 的签名验证器。

In addition, a client can optionally create a replica of the entire database by synchronizing the transaction history from the validators. While creating a replica, a client can verify that validators executed transactions correctly, which increases accountability and transparency in the system. Other clients can read from a client that holds a replica in the same way they would read from a validator to verify the authenticity of the response. For the sake of simplicity, the rest of this paper assumes that clients query a validator directly rather than a replica.
此外，客户端可以通过同步验证器的交易历史来选择创建整个数据库的副本。在创建副本的同时，客户端可以验证验证器是否正确执行了交易，这增加了系统的问责性和透明度。其他客户端可以像从验证器读取副本一样从持有副本的客户端读取，以验证响应的真实性。为了简单起见，本文的其余部分假设客户端直接查询验证器，而不是副本。

Organization. This paper discusses the components of the Libra protocol:
组织。本文讨论了 Libra 协议的组成部分:

- **Logical Data Model** (Section 2) describes the logical data model that organizes the decentralized database visible to validators and clients.

- **逻辑数据模型**（第2节）描述了逻辑数据模型，该模型组织验证器和客户机可见的去中心化数据库。

- **Executing Transactions** (Section 3) describes the use of Move [10], a new programming language that is used to define and execute database transactions.
- **交易执行**(第3节)描述了 Move [10]的使用，这是一种新的编程语言，用于定义和执行数据库事务。

- **Authenticated Data Structures and Storage** (Section 4) describes the mapping of the logical model into authenticated data structures based on Merkle trees [11].
- **认证数据结构和存储**（第4节）描述了逻辑模型基于 Merkle 树映射为认证数据结构的过程[11]。

- **Byzantine Fault Tolerant Consensus** (Section 5) describes the LibraBFT [12] variant of the HotStuff protocol [13], which allows a network with potentially malicious validators to maintain a single, consistent database by executing transactions with Move and coming to agreement on their execution using the authenticated data structures.
- **拜占庭容错共识**(第5节)描述了 Hot Stuff 协议[13]的 LibraBFT [12]变体，该变体允许具有潜在恶意验证者的网络通过使用 Move 执行事务并使用经过认证的数据结构就其执行达成一致来维护单一、一致的数据库。

- **Networking** (Section 6) describes the protocol that enables validators to communicate with each other securely, as required for consensus.
- **网络**（第6节）描述了使验证者能够根据共识的要求安全地相互通信的协议。

Subsequently, we present the open-source Libra Core prototype [6]. Section 7 discusses how Libra Core combines the components of the Libra protocol to process a transaction. Section 8 discusses performance considerations.
随后，我们介绍了开源的 Libra Core 原型[6]。第7节讨论了 Libra Core 如何结合 Libra 协议的组件来处理一笔交易。第8节讨论了性能考虑。

Finally, we explain how the protocol is being used to support the economic stability and governance policies of the Libra ecosystem. Section 9 shows how we use the Move language to implement the low- volatility, reserve-backed Libra coin and a validator management system that mirrors the governance of the Libra Association.
最后，我们解释了该协议如何被用于支持Libra生态系统的经济稳定和治理政策。第9节展示了我们如何使用Move语言来实现低波动性、储备支持的Libra Coin和反映Libra协会治理的验证器管理系统。

Section 10 concludes the paper with a discussion of future plans and ongoing challenges for the Libra ecosystem.
第10节最后讨论了Libra生态系统的未来计划和正在面临的挑战。

2. Logical Data Model

All data in the Libra Blockchain is stored in a single versioned database [14, 15]. A version number is an unsigned 64-bit integer that corresponds to the number of transactions the system has executed. At each version i, the database contains a tuple (T_i, O_i, S_i) representing the transaction (T_i), transaction output (O_i), and ledger state (S_i). Given a deterministic execution function Apply, the meaning of the tuple is: executing transaction Ti against ledger state S_i－1 produces output Oi and a new ledger state Si; that is, Apply(S_i－1, T_i) → ⟨O_i, S_i⟩.
Libra Blockchain 中的所有数据都存储在单个版本的数据库中[14，15]，版本号是一个无符号的64位整数，对应系统已执行的交易数量，在每个版本 i 中，数据库都包含一个元组(T_i, O_i, S_i)，表示交易(T_i)，交易输出(O_i)，和分类账状态(S_i)，给定一个确定性执行函数 Apply ，元组的含义是：执行交易 T_i 对分类账状态 S_i-1产生输出 O_i 和一个新的分类账状态 S_i; 即 Apply (S_i , T_i) → < O_i , S_i> 。

The Libra protocol uses the Move language to implement the deterministic execution function (see Section 3). In this section, we focus on the versioned database, which allows validators to:
Libra 协议使用 Move 语言来实现确定性执行函数(见第3节)。在本节中，我们将重点讨论版本数据库，它允许验证器:

1. Execute a transaction against the ledger state at the latest version.
1.在最新版本中针对分类账状态执行交易。

2. Respond to client queries about the ledger history at both current and previous versions.
2.在当前版本和以前版本中对客户关于分类账历史的查询作出反应。

We first explain the structure of the ledger state stored in a single version and then discuss the purpose of the versioned ledger history view.
我们首先解释存储在单一版本中的分类账状态的结构，然后讨论版本分类账历史视图的目的。

2.1. Ledger State

The ledger state represents the ground truth about the Libra ecosystem, including the quantity of Libra held by each user at a given version. Each validator must know the ledger state at the latest

version in order to execute new transactions.
账本状态代表了Libra生态系统的基本真相，包括每个用户在给定版本中持有的Libra的数量。每个验证者必须知道最新版本中的账本状态，以便执行新的交易。

The Libra protocol uses an account-based data model [16] to encode the ledger state. The state is structured as a key-value store, which maps account address keys to account values. An account value in the ledger state is a collection of published Move resources and modules. The Move resources store data values and modules store code. The initial set of accounts and their state are specified in the genesis ledger state (see Section 3.1).
Libra 协议使用基于账户的数据模型[16]对分类账状态进行编码。状态被结构化为键值对存储，它将账户地址密钥映射到账户值。分类账状态下的账户值是已发布的Move资源和模块的集合。Move资源存储数据值，模块存储代码。初始一组账户及其状态在创世分类账状态中指定(见第3.1节)。

Account addresses. An account address is a 256-bit value. To create a new account, a user first generates a fresh verification/signature key-pair (*vk*,*sk*) for a signature scheme and uses the cryptographic hash of the public verification key *vk* as an account address a = H(*vk*). The new account is created in the ledger state when a transaction sent from an existing account invokes the create_account(a) Move instruction. This typically happens when a transaction attempts to send Libra to an account at address a that has not yet been created.
帐户地址。一个帐户地址是256位值。要创建一个新帐户，用户首先为签名方案生成一个新的验证/签名密钥对(*vk*,*sk*)，*并使用公共验证密钥vk的密码学散列作为帐户地址 a = H(vk). 当从现有帐户发送的交易调用创create_account(a)的Move*指令时，新帐户是在分类账状态下创建的。这通常发生在交易试图将 Libra 发送到尚未创建的地址 a 的帐户时。

Once the new account is created at a, the user can sign transactions to be sent from that account using the private signing key *sk*. The user can also rotate the key used to sign transactions from the account without changing its address, e.g., to proactively change the key or to respond to a possible compromise of the key.
一旦在 a 处创建新账户，用户可以使用私人签署密钥 *sk* 从该账户发送的交易。用户还可以在不改变其地址的情况下旋转用于签署交易的密钥，例如主动改变密钥或响应密钥可能的妥协。

The Libra protocol does not link accounts to a real-world identity. A user is free to create multiple accounts by generating multiple key-pairs. Accounts controlled by the same user have no inherent link to each other. This scheme follows the example of Bitcoin and Ethereum in that it provides pseudonymity [19] for users.
Libra 协议不将账户与现实世界的身份链接。一个用户可以通过生成多个对密钥来自由创建多个账户。由同一个用户控制的账户彼此之间没有固有的链接。该方案遵循了比特币和以太坊的例子，即它为用户

提供了假名[19]。

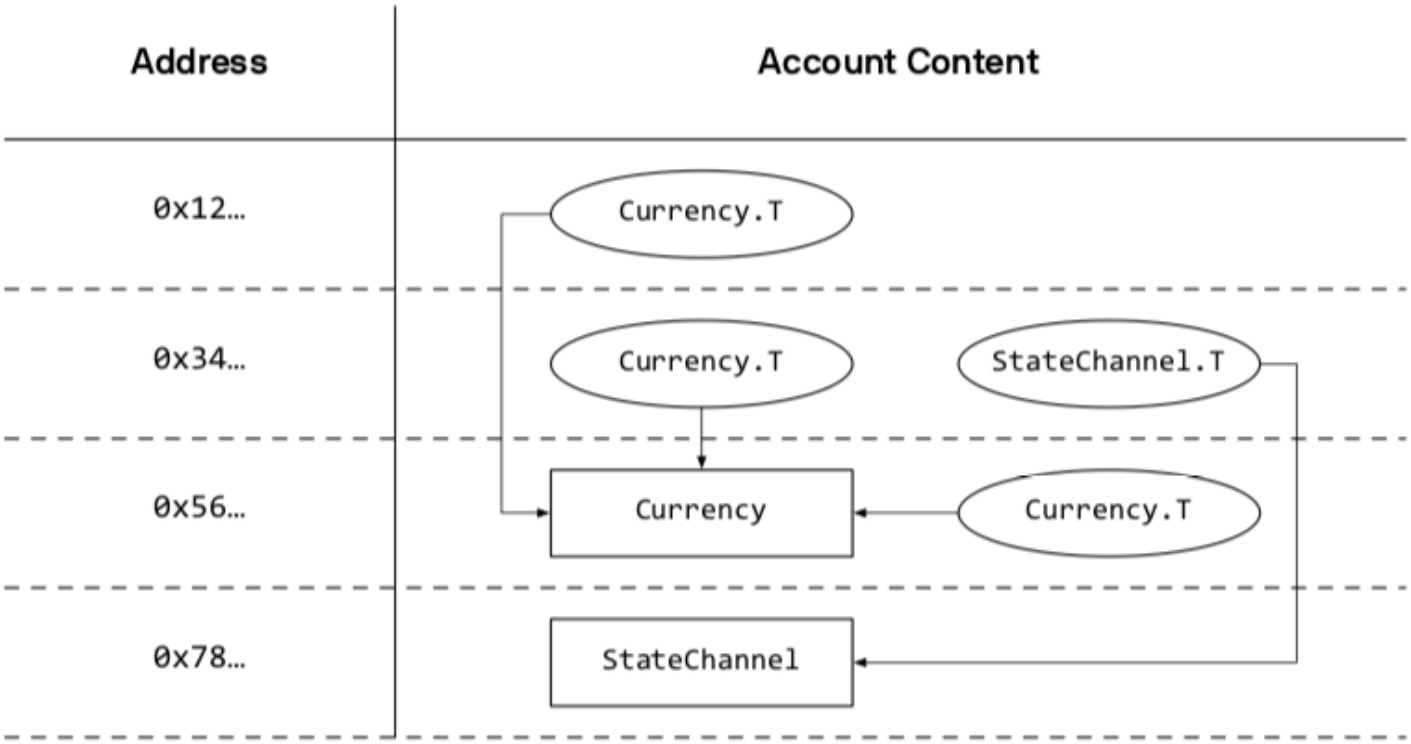

Figure 2: An example ledger state with four accounts. In this diagram, ovals represent resources and rectangles represent modules. A directed edge from a resource to a module means that the type of the resource was declared by that module. The account at address `0x12` contains a `Currency.T` resource declared by the `Currency` module. The code for the `Currency` module is stored at address `0x56`. The account at address `0x34` contains both a `Currency.T` resource and a `StateChannel.T` resource, which is declared by the module stored at address `0x78`.

Resource values. A resource value, or resource, is a record that binds named fields to simple values — such as integers — or complex values — such as other resources embedded inside this resource.

Every resource has a type declared by a module. Resource types are nominal types [20] that consist of the name of the type and the name and address of the resource’ s declaring module. For example, the type of the Currency.T resource in Figure 2 is 0x56.Currency.T. Here, 0x56 is the address where the Currency module is stored, Currency is the name of the module, and Currency.T is the name of the resource.

资源值。资源值或资源，是将命名的字段与简单值（如整数）或复杂值（如嵌入在该资源中的其他资源）绑定的记录。每个资源都有一个由模块声明的类型。资源类型是标称类型[20]，由类型的名称和资源声明模块的名称和地址组成。例如，图2中 Currency.T 资源的类型是0x56.Currency.T。这里，0x56 是存储 Currency 模块的地址， Currency 是模块的名称， Currency.T 是资源的名称。

To retrieve the resource 0x56.Currency.T under account address 0x12, a client would request 0x12/resources/0x56.Currency.T. The purpose of this design is to let modules define a predictable schema for top-level account values — that is, every account stores its 0x56.Currency.T resource under the same path. As such, each account can store at most one resource of a given type. However, this limitation is not restrictive, since programmers can define wrapper resources that organize resources in a custom way (e.g., resource TwoCoin { c1: 0x56.Currency.T, c2: 0x56.Currency.T }).
要在帐户地址0x12下检索资源0x56.Currency.T，客户端将请求0x12/Resource/0x56.Currency.T。该设计的目的是让模块为顶级帐户值定义一个可预测的模式-也就是说，每个帐户都存储其0x56.Currency.T资源在相同的路径。因此，每个帐户最多可以存储一个给定类型的资源。然而，这种限制没有限制性，因为程序员可以定义以自定义方式组织资源的包装器资源(例如，资源 TwoCoin { c1: 0x56.Currency.T, c2: 0x56.Currency.T })。

The rules for mutating, deleting, and publishing a resource are encoded in the module that created the resource and declared its type. Move’ s safety and verification rules prevent other code from making modifications to the resource.
修改、删除和发布资源的规则被编码在创建资源并声明其类型的模块中。 Move 的安全和验证规则阻止其他代码对资源进行修改。

Module values. A module value, or module, contains Move bytecode that declares resource types and procedures (see Section 3.4 for more details). Like a resource type, a module is identified by the address of the account where the module is declared. For example, the identifier for the Currencymodule in Figure 2 is 0x56.Currency.
模块值。模块值或模块包含声明资源类型和过程的Move字节码(详情请参见第3.4节)。与资源类型一样，模块由声明模块的账户地址识别。例如，图2中 Currency 模块的标识符是0x56.Currency 。

A module must be uniquely named within an account — each account can declare at most one module with a given name. For example, the account at address 0x56 in Figure 2 could not declare another module named Currency. On the other hand, the account at address 0x34 could declare a module named Currency. The identifier of this module would be 0x34.Currency. Note that 0x56.Currency.T and 0x34.Currency.T are distinct types and cannot be used interchangeably.
一个模块必须在一个账户中唯一命名——每个账户最多可以用给定的名称声明一个模块。例如，图2中地址0x56的账户不能声明另一个名为货币的模块。另一方面，地址0x34的账户可以声明一个名为货币的模块。该模块的标识符将是0x34。注意，0x56.Currency.T 和0x34.Currency.T 是不同的类型，不能互换使用。

In the current version of the Libra protocol, modules are immutable. Once a module has been declared under an account address, it cannot be modified or deleted, except via a hard fork. We are researching options for a scheme to enable safe module updates in future versions.
在当前版本的 Libra 协议中，模块是不可变的。一旦在账户地址下声明了模块，它就不能被修改或删除，除非通过硬分叉。我们正在研究一种方案的选项，以便在未来的版本中启用安全的模块更新。

2.2. Ledger History

The ledger history stores the sequence of committed and executed transactions as well as the associated events they emitted. The purpose of the ledger history is to keep a record of how the latest ledger state was computed. There is no concept of a block of transactions in the ledger history. The consensus protocol batches transactions into blocks as an optimization and to drive the consensus protocol (see Section 5 for details on consensus). However, in the logical data model, the transactions occur in sequence without distinction as to which block contained each transaction.
账本历史存储已提交和执行交易的序列以及它们发生的相关事件。账本历史的目的是记录最新的账本状态是如何计算的。账本历史中没有一个交易区块的概念。共识协议将交易批量成区块作为优化和驱动共识协议(见第5节关于共识的细节)。然而，在逻辑数据模型中，交易是按顺序发生的，不区分哪个区块包含每个交易。

Although a validator does not need to know the ledger history to execute new transactions, the client can perform authenticated queries against the ledger history and use the ledger history for auditing the transaction execution.
虽然验证器不需要知道分类账历史来执行新交易，但客户端可以针对分类账历史执行经过认证的查询，并使用分类账历史来审计交易的执行。

Responding to client queries. Validators can use the ledger history to answer client queries about previous ledger states, transactions, and outputs. For example, a client might ask about the ledger state at a specific version (e.g., “What was the balance in the account at address x at version 30?”) or the history of events of a certain type (e.g., “What payments did the account at address y receive in the last 20 minutes?”).
回应客户查询。验证者可以使用分类账历史来回答客户关于以前分类账状态、交易和输出的查询。例如，客户可能会询问特定版本的分类账状态(例如，“地址 x 在版本30的账户中的余额是多少？”)或特定类型事件的历史(例如，“地址 y 的账户在过去20分钟收到了什么付款？”)。

Auditing transaction execution. A client can check that the ledger state is correct by re-executing each transaction Ti in the history and comparing the computed ledger state to the corresponding ledger state Si and transaction output Oi in the versioned database. This

mechanism allows clients to audit the validators to ensure that transactions are being executed correctly.
审计交易执行。客户端可以通过重新执行历史中的每个事务 Ti ，并将计算的分类账状态与版本数据库中相应的分类账状态 Si 和事务输出 Oi 进行比较来检查分类账状态是否正确。这种机制允许客户端审计验证器，以确保事务被正确执行。

3. Executing Transactions

In the Libra protocol, the only way to change the blockchain state is by executing a transaction. This section outlines the requirements for transaction execution, defines the structure of transactions, explains how the Move virtual machine (VM) executes a transaction, and describes the key concepts of the Move language.
在 Libra 协议中，改变区块链状态的唯一方法是通过执行交易，本节概述了交易执行的需求，定义了交易的结构，解释了 Move 虚拟机（ VM ）如何执行交易，并描述了 Move 语言的关键概念。

In the initial version of the Libra protocol, only a limited subset of Move’ s functionality is available to users. While Move is used to define core system concepts, such as the Libra currency, users are unable to publish custom modules that declare their own resource types. This approach allows the Move language and toolchain to mature — informed by the experience in implementing the core system components — before being exposed to users. The approach also defers scalability challenges in transaction execution and data storage that are inherent to a general-purpose smart contract platform. As we discuss in Section 10, we are committed to exposing the full programmability supported by the Move language.
在 Libra 协议的初始版本中，只有一个有限的Move功能子集可供用户使用。 虽然 Move 用于定义核心系统概念，如 Libra 货币，但用户无法发布声明自己资源类型的自定义模块。 这种方法允许Move语言和工具链在向用户展示之前成熟——根据实现核心系统组件的经验。 该方法还避开了通用智能合约平台固有的交易执行和数据存储方面的可扩展性挑战。 正如我们在第10节中讨论的那样，我们致力于揭露Move语言支持的全部可编程性。

3.1 Execution Requirements

Known initial state. All validators must agree on the initial, or genesis, ledger state of the system. Because the core components of the blockchain — such as the logic of accounts, transaction validation, validator selection, and Libra coins — are defined as Move modules, the genesis state must define these modules. The genesis state must also have sufficient instantiations of these

core components so that transactions can be processed (e.g., at least one account must be able to pay fees for the first transaction; a validator set must be defined so a quorum of the set can sign the authenticator committing to the first transaction).
已知的初始状态。 所有验证者必须同意系统的初始或起源分类帐状态。 因为区块链的核心组件——如账户的逻辑、交易验证、验证器选择和Libra币——被定义为Move模块，创世状态必须定义这些模块。 创世纪状态还必须有这些核心组件的足够实例，以便能够处理交易(例如，至少一个账户必须能够为第一笔交易支付费用；必须定义验证器集，以便该集的法定人数能够签署承诺第一笔交易的验证器)。

To simplify the design of the system, the initial state of the system is represented as an empty state. The genesis state is then created through a special transaction T0 that defines specific modules and resources to be created, rather than going through the normal transaction process. Clients and validators are configured to accept only ledger histories beginning with a specific T0, which is identified by its cryptographic hash. These special transactions cannot be added to the ledger history through the consensus protocol, only through configuration.
为了简化系统的设计，系统的初始状态被表示为空状态。然后通过定义要创建的特定模块和资源的特殊交易 T0来创建创世纪状态，而不是经历正常的交易过程。客户端和验证器被配置为只接受以特定 T0开头的分类帐历史，该分类帐历史由其密码学哈希来识别。这些特殊交易不能通过共识协议添加到分类帐历史中，只能通过配置。

Deterministic. Transaction execution must be deterministic and hermetic. This means that the output of transaction execution is completely predictable and based only on the information contained within the transaction and current ledger state. Transaction execution does not have external effects (e.g., printing to the console or interacting with the network). Deterministic and hermetic execution ensures that multiple validators can agree on the state resulting from the same sequence of transactions even though transactions are executed independently by each validator. It also means that the transaction history of the blockchain can be re-executed from the genesis block onwards to produce the current ledger state (see Section 2.3).
确定性。 交易执行必须是确定和隐蔽的。 这意味着交易执行的输出是完全可预测的，并且仅基于交易和当前分类账状态中包含的信息。 交易执行不具有外部效果（例如，打印到控制台或与网络交互）。 确定性和隐蔽性执行确保多个验证器可以同意由同一交易序列产生的状态，即使事务由每个验证器独立执行。 这也意味着区块链的交易历史可以从创世纪区块开始重新执行，以产生当前的分类账状态（见第2.3节）。

Metered. In order to manage demand for compute capacity, the Libra protocol charges transaction fees, denominated in Libra coins.5 This follows the gas model popularized by Ethereum [16]. We take the approach of selecting validators with sufficient capacity to meet the needs of the Libra ecosystem (see Section 8). The only intention of this fee is to reduce demand

when the system is under a higher load than it was provisioned for (e.g., due to a denial-of-service attack). The system is designed to have low fees during normal operation, when sufficient capacity is available. This approach differs from some existing blockchains, which target validators with lower capacity and thus at times have more demand to process transactions than throughput. In these systems, fees spike during periods of high demand — representing a revenue source for the validators but a cost for the users.
计价。 为了管理对计算能力的需求，Libra协议收取交易费用，以Libra硬币计价。这遵循了以太坊推广的气体模型[16]。 我们采取的方法是选择有足够能力满足Libra生态系统需求的验证者(见第8节)。 这一费用的唯一目的是在系统承受的负载高于其预定负载时(例如，由于拒绝服务攻击)，减少需求。 该系统的设计是在正常运行时，当有足够的容量时，收费很低。 这种方法不同于一些现有的区块链，这些区块链针对的是容量较低的验证器，因此有时处理交易的需求多于吞吐量。 在这些系统中，在高需求时期，费用会飙升——这代表了验证者的收入来源，但对于用户是成本。

The size of the fee is determined by two factors: gas price and gas cost. Each transaction specifies a price in Libra per unit of gas that the submitter is willing to pay. The execution of a transaction dynamically accounts for the computational power it expends in the form of a gas cost. Validators prioritize executing transactions with higher gas prices and may drop transactions with low prices when the system is congested. This reduces the demand for transactions under high load.
费用的大小由燃气价格和燃气成本两个因素决定，每笔交易都规定了提交者愿意支付的单位燃气中的Libra价格，交易的执行动态核算其以燃气成本的形式消耗的算力，验证者优先执行燃气价格较高的交易，在系统拥堵时可能会丢弃价格较低的交易，这降低了高负荷下交易的需求。

In addition, a transaction includes a maximum gas amount, which specifies the maximum number of gas units that the submitter is willing to pay for at the specified price. During execution, the VM tracks the number of gas units used. If the maximum gas limit is reached before execution completes, the VM halts immediately. None of the partial changes resulting from such a transaction are committed to the state. However, the transaction still appears in the transaction history, and the sender account is charged for the gas used.
此外，交易还包括一个最大气体量，它指定了提交者愿意以指定价格支付的最大气体单元数量，在执行过程中， VM 跟踪使用的气体单元数量，如果在执行完成前达到了最大气体限值， VM 立即停止，这样的交易产生的部分更改都没有被承诺到状态中，但交易仍然出现在交易历史中，发送者账户对使用的气体进行收费。

As we discuss in this section, many parts of the core logic of the blockchain are defined using Move, including the deduction of gas fees. To avoid circularity, the VM disables the metering of gas during the execution of these core components. These core components must be defined in the genesis state and must be written defensively to prevent malicious transactions from triggering

the execution of computationally expensive code. The cost of executing this logic can be included in the base fee charged for the transaction.
正如我们在本节中讨论的那样，区块链核心逻辑的许多部分都是使用 Move 来定义的，包括扣除燃气费，为了避免循环性， VM 在执行这些核心组件时禁用燃气的计量，这些核心组件必须在创世状态下定义，必须防御性地编写，以防止恶意交易触发计算昂贵代码的执行，执行这种逻辑的成本可以计入交易收取的基础费用。

Asset semantics. As we explain in Section 2.1, the ledger state directly encodes digital assets with real-world value. Transaction execution must ensure that assets such as Libra coins are not duplicated, lost, or transferred without authorization. The Libra protocol uses the Move virtual machine (Section 3.4) to implement transactions and custom assets with these properties safely.
资产语义。正如我们在第2.1节中解释的那样，账本状态直接编码具有现实世界价值的数字资产。交易执行必须确保Libra币等资产不被复制、丢失或擅自转移。Libra协议使用 Move 虚拟机（第3.4节）安全地实现具有这些属性的交易和自定义资产。

3.2. Transaction Structure

A transaction is a signed message containing the following data:
交易是包含下列数据的签名消息：

- **Sender address:** The account address of the transaction sender. The VM reads the sequence number, authentication key, and balance from the Libra Account.T resource stored under this address.
- **发件人地址：**交易发件人的帐户地址。虚拟机从存储在该地址下的 Libra Account.T资源读取序列号、身份验证密钥和余额。

- **Sender public key:** The public key that corresponds to the private key used to sign the transaction. The hash of this public key must match the authentication key stored under the sender’ s LibraAccount.T resource.
- **发件人公钥:** 对应于用于签署交易的私钥的公钥。该公钥的哈希必须与发件人存储在发件人的 LibraAccount.T 资源下的身份验证密钥相匹配。

- **Program:** A Move bytecode transaction script to execute, an optional list of inputs to the script, and an optional list of Move bytecode modules to publish.
- **程序：**需要执行的Move字节码交易脚本、脚本输入的可选列表和需要发布的Move字节码模块的可选列表。

- **Gas price:** The number of Libra coins that the sender is willing to pay per unit of gas in order to execute this transaction.
- **燃气价格：**发送者为了执行本次交易，愿意支付单位燃气的Libra数量。

- **Maximum gas amount:** The maximum number of gas units that the transaction is allowed to consume before halting.
- **最大气量：**交易停止前允许消耗的最大气量。

- **Sequence number:** An unsigned integer that must be equal to the sequence number from the sender’ s LibraAccount.T resource. After this transaction executes, the sequence number is incremented by one. Since only one transaction can be committed for a given sequence number, transactions cannot be replayed.
- **序列号:** 一个无符号的整数，必须等于来自发送方的 LibraAccount.T 资源的序列号。此交易执行后，序列号被增加一个。由于只能为给定的序列号提交一个交易，交易不允许重放。

3.3. ExecutingTransactions

Executing a transaction proceeds through a sequence of six steps inside the VM. Execution is separate from the update of the ledger state. First, a transaction is executed as part of an attempt to reach agreement on its sequencing. Since the execution is hermetic, this can be done without causing external side effects. Subsequently, if agreement is reached, its output is written to the ledger history. Executing a transaction performs the following six steps:
执行交易通过虚拟机内的六个步骤顺序进行。执行与分类账状态的更新是分开的。首先，执行交易是试图就其顺序达成协议的一部分。由于执行是隐蔽的，这可以在不引起外部副作用的情况下完成。随后，如果达成协议，它的输出将写入分类账历史。执行事务执行以下六个步骤:

(1) Check signature. The signature on the transaction must match the sender’ s public key and the transaction data. This step is a function only of the transaction itself — it does not require reading any data from the sender’ s account.
（1）检查签名。交易上的签名必须与发送方的公钥和交易数据相匹配。这一步仅是交易本身的一个函数--它不需要从发送方的账户中读取任何数据。

(2) Run prologue. The prologue authenticates the transaction sender, ensures that the sender has sufficient Libra coin to pay for the maximum number of gas units specified in the transaction, and checks that the transaction is not a replay of a previous transaction. All of these checks are

implemented in Move via the prologue procedure of the LibraAccount module. Gas metering is disabled during the execution of the prologue. Specifically, the prologue does the following:
(2)运行序幕。序幕认证交易发件人，确保发件人有足够的Libra硬币支付交易中指定的最大数量的气体单位，并检查交易不是前一次交易的重放。所有这些检查都通过 LibraAccount 模块的序幕程序在Move中实现。序幕执行期间禁用气体计量。具体来说，序幕做以下工作:

- Checks that the hash of the sender’ s public key is equal to the authentication key stored under the sender’ s account. Without this check, the VM would erroneously accept a transaction with a cryptographically valid signature even though there is no correspondence to the key that is associated with the account.
- 检查发送方的公共密钥的哈希等于存储在发送方账户下的验证密钥。如果没有这个检查，虚拟机将错误地接受具有密码有效签名的交易，即使与账户相关的密钥没有对应关系。

- Checks that gas_price * max_gas_amount <= sender_account_balance. Without this check, the VM would execute transactions that may fail in the epilogue because they would be unable to pay for gas.
- 检查 gas_price * max_gas_amount <= sender_account_balance。如果没有这个检查， VM 将执行在结语中可能失败的交易，因为他们无法支付燃气。

- Ensure that the transaction sequence number is equal to the sequence number stored under the user’ s account. Without this check, an adversary could replay old transactions (e.g., Bob could replay a transaction from Alice that paid him ten Libra coins).
- 确保交易序列号等于用户账户下存储的序列号。如果没有这个检查，对手可以重放旧的交易(例如，鲍勃可以重新播放爱丽丝支付给他10个Libra硬币的交易)。

(3) Verify transaction script and modules. Once the transaction prologue has completed success- fully, the VM performs well-formedness checks on the transaction script and modules using the Move bytecode verifier. Before actually running or publishing any Move code, the bytecode verifier checks crucial properties like type-safety, reference-safety (i.e., no dangling references), and resource-safety
(i.e., resources are not duplicated, reused, or inadvertently destroyed).
(3)验证事务脚本和模块。一旦交易序幕圆满完成，虚拟机使用 Move 字节码验证器对交易脚本和模块进行良好的格式检查。在实际运行或发布任何 Move 代码之前，字节码验证器检查关键的属性，如类型安全、引用安全(即没有悬挂引用)和资源安全(即资源不被重复、重复使用或无意中销毁)。

(4) Publish modules. Each module in the program field of the transaction is published under the transaction sender’ s account. Duplicate module names are prohibited — for example, if the

transaction attempts to publish a module named M to an account that already contains a module named M, the step will fail.
(4)发布模块。交易程序字段中的每个模块都在交易发送者的账户下发布。禁止重复的模块名称-例如，如果交易试图将一个名为 M 的模块发布到已经包含一个名为 M 的模块的账户，该步骤将失败。

(5) Run transaction script. The VM binds the transaction arguments to the formal parameters of the transaction script and executes it. If this script execution completes successfully, the write operations performed by the script and the events emitted by the script are committed to the global state. If the script execution fails (e.g., due to having run out of gas or a runtime execution failure), no changes from the script are committed to the global state.
(5)运行交易脚本。 VM 将交易参数与交易脚本的形式参数绑定并执行它。如果这个脚本执行成功，脚本执行的写入操作和脚本发出的交易将被承诺到全局状态。如果脚本执行失败(例如，由于气体用完或运行时执行失败)，则不将脚本的任何更改承诺到全局状态。

(6) Run epilogue. Finally, the VM runs the transaction epilogue to charge the user for the gas used and increment the sender’ s account sequence number. Like the prologue, the transaction epilogue is a procedure of the Move LibraAccount module and runs with gas metering disabled. The epilogue is always run if execution advances beyond step (2), including when steps (3), (4), or (5) fail. The prologue and the epilogue work together to ensure that all transactions accepted in the ledger history are charged for gas. Transactions that do not proceed beyond step (2) are not appended to the ledger history. The fact that these transactions were considered for execution is never recorded. If a transaction advances past step (2), the prologue has ensured that the account has enough Libra coins to pay for the maximum number of gas units allowed for the transaction. Even if the transaction runs out of gas, the epilogue is able to charge it for this maximum amount.
（6）运行结语。 最后， VM 运行交易结语，向用户收取所用气体的费用，并增加发送方的帐户序列号。 与序幕一样，交易结语是 Move的LibraAccount模块的一个过程，并禁用气体计量。 如果执行超过步骤(2)，包括步骤(3)、(4)或(5)失败时，结语总是运行。 序幕和结语一起工作，以确保分类账历史上接受的所有交易都按气体收费。 没有超越步骤(2)进行的交易不附加到分类账历史记录中。 这些交易被考虑执行的事实从未被记录。 如果交易推进到第(2)步，序幕已经确保账户有足够的天秤币来支付交易允许的最大气体单位数量。 即使交易耗尽了汽油，结语也能为这个最大金额收取费用。

3.4 The Move Programming Language

As we have seen, a transaction is an authenticated wrapper around a Move bytecode program. Move [10] is a new programming language created during the design of the Libra protocol. Move has three important roles in the system:

正如我们所看到的，交易是一个围绕Move字节码程序的经过认证的包装程序。Move[10]是在设计 Libra 协议期间创建的一种新的编程语言。Move在系统中有三个重要角色:

1. To enable flexible transactions via transaction scripts.
1.通过交易脚本启用灵活的交易。

2. To allow user-defined code and datatypes, including “smart contracts” via modules.
2.通过模块允许用户定义代码和数据类型，包括“智能合约”。

3. To support configuration and extensibility of the Libra protocol (see Section 9).
3.支持 Libra 协议的配置和可扩展性（见第9节）

The key feature of Move is the ability to define custom resource types, which have semantics inspired by linear logic [21]. Resource types are used to encode programmable assets that behave like ordinary program values: resources can be stored in data structures, passed as arguments to procedures, and so on. However, the Move type system provides special safety guarantees for resources. A resource can never be copied, only moved. In addition, a resource type can only be created or destroyed by the module that declares the type. These guarantees are enforced statically by the Move VM. This allows us to represent Libra coins as a resource type in the Move language (in contrast to Ether and Bitcoin, which have a special status in their respective languages).
Move 的关键特征是定义自定义资源类型的能力，这些资源类型具有受线性逻辑启发的语义[21]。 资源类型用于编码行为类似于普通程序值的可编程资产：资源可以存储在数据结构中，作为参数传递给程序，等等。 然而， Move 类型的系统为资源提供了特殊的安全保障。 一个资源永远不可能被复制，只有被搬运。 此外，资源类型只能由声明类型的模块创建或销毁。 这些保证由Move VM 静态执行。 这使得我们可以在 Move 语言中将Libra币表示为一种资源类型（与 Ether 和比特币形成对比，它们在各自语言中具有特殊地位）。

We have published a separate, more detailed report on the design of Move [10]. In the remainder of this section, we give a brief overview of the key Move concepts for transaction execution: developing Move transaction scripts and modules and executing Move bytecode with the virtual machine.
我们发表了一份单独的、更详细的关于 Move [10]设计的报告。在本节的其余部分，我们简要概述了交易执行的关键 Move 概念:开发 Move 交易脚本和模块，并用虚拟机执行 Move 字节码。

Writing Move programs. There are three different representations of a Move program: source code, intermediate representation (IR), and bytecode. We are currently in the process of designing

the Move source language, which will be an ergonomic language designed to make it easy to write and verify safe code. In the meantime, programmers can develop modules and transaction scripts in Move IR. Move IR is high level enough to write human-readable code, yet low level enough to directly translate to Move bytecode. Both the future Move source language and the Move IR are compiled into Move bytecode, which is the format used by the Libra protocol.
写移动程序。 Move程序有三种不同的表示:源代码、中间表示(IR)和字节码。 我们目前正在设计Move源语言，这将是一种符合人体工程学的语言，旨在使其易于编写和验证安全代码。 与此同时，程序员可以在 Move IR 中开发模块和交易脚本。 Move IR 的级别足够高，可以编写人类可读的代码，但低到可以直接转换为 Move 字节码。 未来的Move源语言和移动 IR 都被编译成Move字节码，这是 Libra 协议使用的格式。

We use a verifiable bytecode as the executable representation of Move for two reasons:
我们使用可验证的字节码作为Move的可执行表示，原因有二：

- The safety guarantees described above must apply to all Move programs. Enforcing these guarantees in the compiler is not enough. An adversary could always choose to bypass the compiler by writing malicious code directly using the bytecode (running a compiler as part of transaction execution would make execution slower and more complex and would require validators trusting the correctness of the full compiler code base). Thus, the protocol avoids trusting the compiler by enforcing all of Move’ s safety guarantees via bytecode verification: type-safety, reference-safety, and resource-safety.
- 上面描述的安全保证必须适用于所有Move程序。在编译器中强制执行这些保证是不够的。对手可以选择绕过编译器，直接使用字节码编写恶意代码(作为交易执行的一部分运行编译器会使执行变得更慢和更复杂，并需要验证者信任整个编译器代码库的正确性)。因此，协议通过字节码验证来执行所有Move的安全保证(类型安全、引用安全和资源安全)，避免了信任编译器。

- Move’ s stack-based bytecode has fewer instructions than a higher-level source language would. In addition, each instruction has simple semantics that can be expressed via an even smaller number of atomic steps. This reduces the specification footprint of the Libra protocol and makes it easier to spot implementation mistakes.
- Move基于堆栈的字节码比更高级别的源语言拥有更少的指令。此外，每个指令都有简单的语义，可以通过更小的原子步骤来表达。这减少了 Libra 协议的规范足迹，更容易发现实现错误。

Transaction scripts. A transaction script is the main procedure of the Libra protocol. Transaction scripts enable flexible transactions because a script is an arbitrary Move bytecode program. A script can invoke multiple procedures of modules published in the ledger state, use conditional

logic, and perform local computation. This means that scripts can perform expressive one-off actions, such as paying a specific set of recipients.
事务**脚本**。交易脚本是 Libra 协议的主要程序。交易脚本执行灵活的交易，因为一个脚本是任意Move字节码程序。一个脚本可以调用在分类账状态下发布的模块的多个程序，使用条件逻辑，并执行局部计算。这意味着脚本可以执行表达式的一次性操作，例如支付特定的一组收件人。

We expect that most transaction scripts will perform a single procedure call that wraps generic func- tionality. For example, the LibraAccount.pay_from_sender(recipient_address, amount) procedure encapsulates the logic for performing a peer-to-peer payment. A transaction script that takes recipient_address and amount as arguments and invokes this procedure is the same as an Ether transfer transaction in Ethereum [16].
我们预计大多数交易脚本将执行一个包含通用功能性的单一过程调用。例如，LibraAccount.pay_from_sender(recipient_address, amount) 程序封装了执行点对点支付的逻辑。一个以收件人地址和金额为参数并调用该程序的交易脚本与以太坊中的以太转账交易相同[16]。

Modules. A module is a code unit published in the ledger state. A module declares both struct types and procedures. A struct value contains data fields that may hold primitive values, such as integers or other struct values.
模块。模块是在分类账状态下发布的代码单元。模块声明结构类型和过程。结构值包含可能包含原始值的数据字段，如整数或其他结构值。

Each struct must be tagged as either resource or unrestricted (i.e., non-resource). Unrestricted structs are not subject to the restrictions on copying and destruction described above. However, unrestricted structs cannot contain resource structs (either directly or transitively) and cannot be published under an account in the ledger state.
每个结构都必须标记为资源或无限制结构(即非资源)。无限制结构不受上述复制和销毁的限制。但是，无限制结构不能包含资源结构(直接或短暂)，也不能在分类账状态下的账户下发布。

At a high level, the module/struct/procedure relationship is similar to the class/object/method rela- tionship in object-oriented programming. However, there are important differences. A module may declare multiple struct types (or zero struct types). No data field can be accessed outside of its declaring module (i.e., no public fields). The procedures of a module are static procedures rather than instance methods; there is no concept of this or self in a procedure. Move modules are similar to a limited version of ML-style modules [22] without higher-order functions.
在高层次上，模块/结构体/过程关系类似于面向对象编程中的类/对象/方法放松关系。然而，这有重要的区别。一个模块可以声明多个结构体类型(或零结构体类型)。在其声明模块之外不能访问任何数据字

段(即没有公共字段)。一个模块的过程是静态过程，而不是实例方法；在一个过程中没有这个或自我的概念。Move模块类似于 ML风格模块的有限版本[22]，没有更高阶的函数。

Move modules are related to, but not the same as, the concept of “smart contracts” in Ethereum and other blockchain platforms. An Ethereum smart contract contains both code and data published in the ledger state. In Libra, modules contain code values, and resources contain data values. In object-oriented terms, an Ethereum smart contract is like a singleton object published under a single account address. A module is a recipe for creating resources, but it can create an arbitrary number of resources that can be published under different account addresses.
Move模块与以太坊和其他区块链平台中的“智能合约”概念有关，但与之不一样，一个以太坊智能合约既包含代码，也包含在账本状态下发布的数据，在 Libra 中，模块包含代码值，资源包含数据值，在面向对象的术语中，以太坊智能合约就像一个在单个账号地址下发布的单个对象，一个模块是创建资源的配方，但它可以创建一个任意数量的资源，可以在不同的账号地址下发布。

The Move virtual machine. The Move virtual machine implements a verifier and an interpreter for the Move bytecode. The bytecode targets a stack-based virtual machine with a procedure-local operand stack and registers. Unstructured control-flow is encoded via gotos and labels.
Move虚拟机。Move虚拟机实现了Move字节码的验证器和解释器。字节码以基于堆栈的虚拟机为目标，具有过程本地操作数堆栈和寄存器。非结构化控制流通过 gotos 和标签编码。

A developer writes a transaction script or module in Move IR that is then compiled into the Move bytecode. Compilation converts structured control-flow constructs (e.g., conditionals, loops) into unstructured control-flow and converts complex expressions into a small number of bytecode instruc- tions that manipulate an operand stack. The Move VM executes a transaction by verifying then running this bytecode.
开发人员在移动 IR 中编写一个交易脚本或模块，然后编译到Move字节码中。编译将结构化的控制流结构(例如，条件、循环)转换为非结构化的控制流，并将复杂的表达式转换为少量的字节码指令，这些指令操纵操作数堆栈。Move VM 通过验证然后运行这个字节码来执行交易。

The Move VM supports a small number of types and values: booleans, unsigned 64-bit integers, 256-bit addresses, fixed-size byte arrays, structs (including resources), and references. Struct fields cannot be reference types, which prevents the storage of references in the ledger state.
Move VM 支持少量的类型和值： booleans 、无符号的64位整数、256位地址、固定大小的字节数组、结构体（包括资源）和引用。结构体字段不能是引用类型，这会阻止引用在分类账状态下的存储。

The Move VM does not have a heap — local data is allocated on the stack and freed when the allocating procedure returns. All persistent data must be stored in the ledger state.

Move VM 没有堆-本地数据在堆栈上分配，当分配过程返回时释放。所有持久数据必须存储在分类账状态中。

4. Authenticated Data Structures and Storage

After executing a transaction, a validator translates the changes to the logical data model into a new version of an authenticated data structure [7, 8, 9] used to represent the database. The short authenticator of this data structure is a binding commitment to a ledger history, which includes the newly executed transaction.
在执行交易后，验证器将对逻辑数据模型的更改转化为用于表示数据库的经过认证的数据结构[7，8，9]的新版本，该数据结构的短验证器是对账本历史的绑定承诺，其中包括新执行的交易。

Like transaction execution, the generation of this data structure is deterministic. The consensus protocol uses this authenticator to agree on an ordering of transactions and their resulting execution (we discuss consensus in detail in Section 5). As part of committing a block of transactions, validators
collectively sign the short authenticator to the new version of the resulting database.
和交易执行一样，这种数据结构的生成是确定性的。共识协议使用这个验证器来商定交易的顺序及其结果的执行(我们在第5节中详细讨论了共识)。作为提交事务块的一部分，验证器集体将短认证器签名到所产生的数据库的新版本。

Using this collective signature, clients can trust that a database version represents the full, valid, and irreversible state of the database’ s ledger history. Clients can query any validator (or a third-party replica of the database) to read a specific database value and verify the result using the authenticator and a short proof. Therefore, clients do not need to trust the party that executes the query for the correctness of the resulting read.
使用这种集体签名，客户端可以相信数据库版本代表数据库分类账历史的完整、有效和不可逆状态。客户端可以查询任何验证器(或数据库的第三方复制品)来读取特定的数据库值，并使用验证器和简短证明验证结果。因此，客户端不需要信任执行查询的一方，以获得读取的正确性。

Data structures in the Libra protocol are based on Merkle trees and inspired by those of other blockchains; however, in several cases, we have made slightly different decisions which we highlight below. First, we briefly discuss the Merkle tree approach to creating an authenticator. We then describe the authenticated data structure, starting from the root of the data structure, then we discuss the substructures within it. Figure 3 depicts the data structure and provides a visual guide to this section.

Libra 协议中的数据结构基于 Merkle 树，并受其他区块链的启发；然而，在几个案例中，我们做出了略有不同的决定，我们在下面强调。首先，我们简单讨论了 Merkle 树创建身份验证器的方法。我们接着描述了经过认证的数据结构，从数据结构的根开始，然后我们讨论了其中的子结构。图3描述了数据结构，并为本节提供了可视化指南。

4.1. Background on Authenticated Data Structures

An authenticated data structure allows a verifier V to hold a short authenticator a, which forms a binding commitment to a larger data structure D. An untrusted prover P, which holds D, computes f(D) → r and returns both r — the result of the computation of some function f on D — as well as π — a proof of the correct computation of the result — to the verifier. V can run Verify(a, f, r, π), which returns true if and only if f(D) = r. In the context of the Libra Blockchain, provers are generally validators and verifiers are clients executing read queries. However, clients — even those with only a partial copy of the database — can also serve as a prover and perform authenticated read queries for other clients.
一个授权的数据结构允许一个验证器V持有一个短的授权a, 这个短的授权a 对于一个大的数据结构D形成了一个承诺绑定。一个不可信的证明这 P，它可能持有D，计算f(D) → r。并返回r 和 π， 其中r 是函数f在D上执行的结果；π 对于验签器来说是计算结果正确的证明。V 可以通过运行Verify(a, f, r, π)， 当切仅当f(D) = r 才返回真。在 Libra Blockchain 的上下文中，验证者通常是验证者，验证者是执行读取查询的客户端。然而，客户端——即使是那些只有部分数据库副本的客户端——也可以作为验证者，为其他客户端执行经过认证的读取查询。

Merkle trees [11] are a common form of authenticated data structure, used to store maps between integers and string values. In a Merkle tree of size 2k, the structure D maps every integer key i ∈ [0,2k) to a string value si. The authenticator is formed from the root of a full binary tree created from the strings, labeling leaves as H(i ∥ si) and internal nodes as H(left || right), where H is a cryptographic hash function (which we will refer to as a hash). The function f, which the prover wishes to authenticate, is an inclusion proof that a key-value pair (k, v) is within the map D.
Merkle tree [11]是一种常见的认证数据结构形式，用于存储整数值和字符串值之间的映射。 在大小为 2k 的 Merkle 树中，结构 D 将每个整数键i ∈ [0,2k) 映射到字符串值 si 。 验证器是由字符串创建的全二叉树的根部形成的，将叶子标记为 H(i || si)，内部节点标记为 H(left || right)，其中 H 是密码学散列函数(我们将其称为散列函数)。验证者希望验证的函数 f 是包含证明一个键值对(k, v)在映射 D 中。

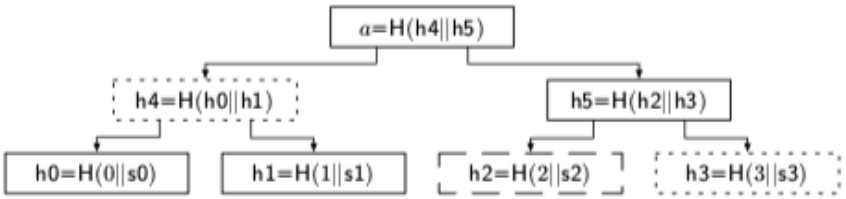

Figure 4: A Merkle tree storing $D = \{0 : \mathsf{s0}, \ldots\}$. If f is a function that gets the third item (shown with a dashed line) then $r = \mathsf{s2}$ and $\pi = [\mathsf{h3}, \mathsf{h4}]$ (these nodes are shown with a dotted line). $\mathsf{Verify}(a, f, r, \pi)$ verifies that $a \stackrel{?}{=} \mathsf{H}(\mathsf{h4} \| \mathsf{H}(\mathsf{H}(2 \| r) \| \mathsf{h3}))$.

P authenticates lookups for an item i in D by returning a proof π that consists of the labels of the sibling of each of the ancestors of node i. Figure 4 shows a lookup for item three in a Merkle tree of size four. Item three’ s node is shown with a dotted line, and the nodes included in π are shown with a dashed line.
P 通过返回一个由节点 i 祖先的兄弟姐妹的标签组成的证明π来验证 D 中项目 i 的查找。图4显示了在尺寸为4的 Merkle 树中对项目3的查找。项目3的节点用虚线显示，π中包含的节点用虚线显示。

4.2. Ledger History

Most blockchains, starting with Bitcoin [24], maintain a linked list of each block of transactions agreed on by the consensus protocol with a block containing the hash of a single ancestor. This structure leads to inefficiencies for clients. For example, a client that trusts some block B and wants to verify information in an ancestor block B′ needs to fetch and process all intermediate ancestors.
大多数区块链，从比特币[24]开始，用包含单个祖先的哈希的区块来维护共识协议约定的每一个交易区块的链表，这种结构导致客户端的效率低下，例如，一个客户端信任某些区块 B ，并希望验证某个祖先区块 B ′中的信息，需要提取和处理所有中间祖先。

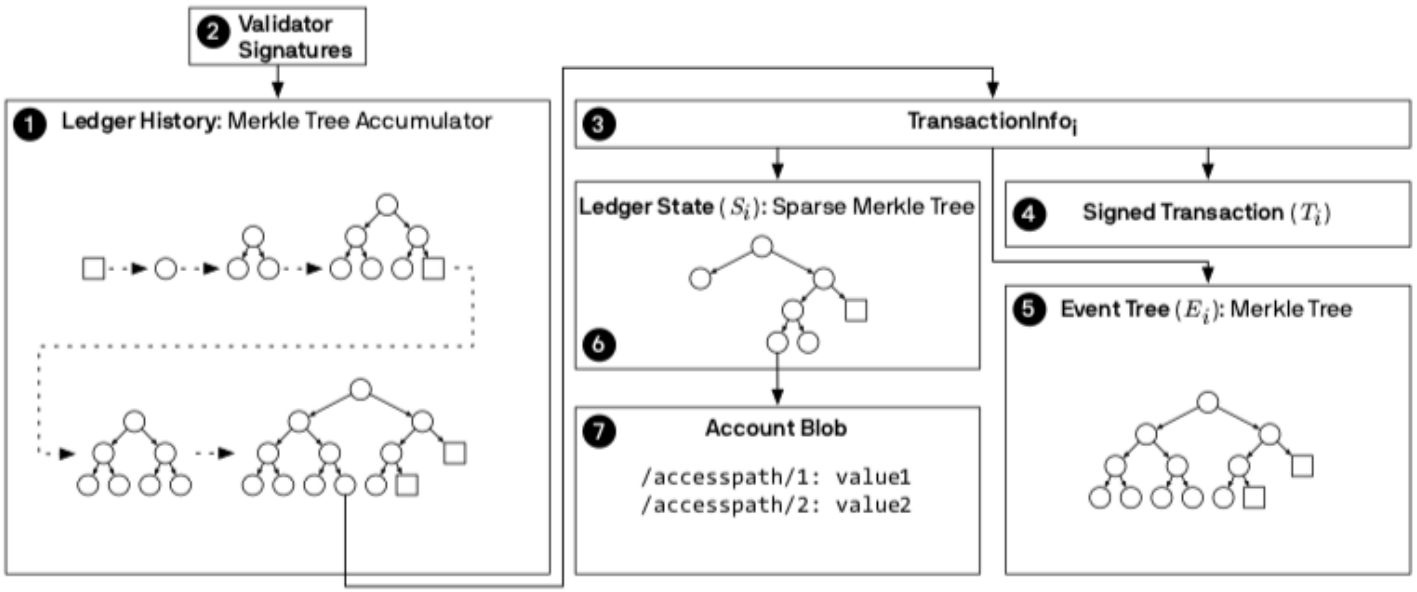

Figure 3: (❶) The root hash of the ledger history structure is the authenticator to the full state of the system that is (❷) signed by a quorum of validators. As transactions are added to the database, the authenticator (Section 4.2) committing to the ledger history grows (depicted with dashed arrows). Each leaf of the ledger history commits to a (❸) TransactionInfo structure. This structure commits to the (❹) signed transaction (T_i), (❺) the list of events generated during that transaction (E_i, Section 4.5), and the (❻) state after the execution of that transaction (S_i, Section 4.3). The state is a sparse Merkle tree with an (❼) account blob at each leaf.

The Libra protocol uses a single Merkle tree to provide an authenticated data structure for the ledger history. Figure 3 illustrates the full data structure that underpins the Libra database, including the ledger history that contains TransactionInfo_i records, which, in turn, contain information about database states, events, and accounts. The ledger history is represented as a Merkle tree, mapping a sequential database version number i to a TransactionInfoi structure. Each TransactionInfoi structure contains a signed transaction (Ti), the authenticator for the state after the execution of Ti (Si, discussed in Section 4.3), and the authenticator for the events generated by Ti (Ei, discussed in Section 4.5). Like other Merkle trees, the ledger history supports O(log n)-sized proofs — where n is the total number of transactions processed — to authenticate the lookup of a specific TransactionInfoi. When a client wishes to query the state of version i or lookup an event generated in version i, it performs an authenticated lookup of TransactionInfoi along with an authenticated lookup using the contained state or event list authenticator.
Libra 协议使用单个 Merkle 树为分类账历史提供经过认证的数据结构。 图3展示了支撑 Libra 数据库的完整数据结构，包括包含 TransactionInfo_i 记录的分类账历史，这些记录反过来又包含关于数据库状态、事件和账户的信息。 分类账历史被表示为 Merkle 树，将顺序数据库版本号 i 映射到 Transaction Info_i 结构。 每个 TransactionInfo_i 结构都包含一个签名交易(T_i)， T_i (S_i ，在第4.3节中讨论)执行后状态的验证器，以及 T_i (E_i ，在第4.5节中讨论)生成的事件的验证器。 像其他 Merkle 树一样，分类账历史支持 O (log n)大小的证明，其中 n 是处理的交易总数，以验证对特定 TransactionInfo_i 的

查找。 当客户端希望查询版本 i 的状态或查找版本 i 中生成的事件时，它会使用包含的状态或事件列表验证器执行对 TransactionInfo_i 的身份验证查找以及身份验证查找。

Specifically, the ledger history uses the Merkle tree accumulator [25, 26] approach to form Merkle trees, which also provides efficient append operations. This operation is useful, as a ledger history can be incrementally computed by appending new transactions to an old ledger history. Merkle tree accumulators can also generate an efficient proof that, given an authenticator a committing to the ledger info up to transaction i, an authenticator a′ committing to the ledger info up to j > i has an identical history up to transaction i — in other words, proving that a is a prefix of a′. These proofs allow efficient verification that one ledger history commitment is a continuation of another.
具体来说，账本历史使用 Merkle 树累加器[25，26]方法来形成 Merkle 树，这也提供了高效的附加操作。 这种操作是有用的，因为账本历史可以通过在旧的账本历史上附加新交易来递增计算。 Merkle 树累加器还可以生成一个有效的证明，即给定验证器 a '向账本信息承诺到交易 i ，验证器 a '向账本信息承诺到 j > i ，在交易 i 之前有一个相同的历史——换句话说，证明 a 是 a '的前缀。 这些证明可以有效地验证一个分类账历史承诺是另一个分类账的延续。

Pruning Storage. The root hash of the ledger history Merkle accumulator is an authenticator for the full state of the system. It commits to the state at every existing version of the system, every transaction ever sent, and every event ever generated. While the state storage described in Section 4.3 allows efficient storage of multiple versions of the ledger state, validators may wish to reduce the space consumed by past versions. Validators are free to prune old states as they are not necessary for the processing of new transactions. Merkle tree accumulators support the pruning of historical data, requiring only O(log n) storage to append new records [25].
修剪仓库 分类账历史 Merkle 累加器的根哈希是系统全部状态的验证器。 它承诺在系统的每个现有版本、发送的每个交易和生成的每个交易中的状态。 虽然第4.3节中描述的状态存储允许高效存储分类账状态的多个版本，但验证器可能希望减少过去版本消耗的空间。 验证者可以自由修剪旧状态，因为旧状态对于处理新交易来说并不是必要的。 Merkle 树蓄类积器支持对历史数据的修剪，只需要 O （ log n ）存储即可附加新记录[25]。

Any replica of the system is free to retain the entire state and can trace the authenticity of their data to the root hash signed by the consensus protocol. Replicas are more amenable to scaling than validators. Replicas do not need to be secured as they do not participate in consensus, and multiple replicas can be created to handle read queries in parallel.
系统的任何副本都可以自由保留整个状态，并可以将其数据的真实性追溯到共识协议签署的根哈希。复制器比验证器更易于伸缩。复制器不需要得到安全保护，因为它们不参与共识，可以创建多个复制器来并行处理读查询。

4.3. Ledger State

A ledger state S_i represents the state of all accounts at version i as a map of key-value pairs. Keys are based on the 256-bit account addresses, and their corresponding value is the authenticator of the account (discussed in Section 4.4).7 Using a representation similar to Figure 4, the map is represented as a Merkle tree of size 2^256.
分类账状态 S_i 表示版本 i 所有账户的状态，作为密钥值对的映射。密钥基于256位账户地址，其对应的值是账户的认证者(在第4.4节中讨论)。7使用类似于图4的表示，该映射被表示为大小为2^256的 Merkle 树。

While a tree of size 2^256 is an intractable representation, optimizations can be applied to form a sparse Merkle tree. Figure 5 shows two optimizations that are applied to transform a naive implementation (1) into an efficient one. First, subtrees that consist entirely of empty nodes are replaced with a placeholder value (2) as used in the certificate transparency system [27]. This optimization creates a representation of a tractable size without substantially changing the proof generation of the Merkle tree. However, leaves are always stored at the bottom level of the tree, meaning that the structure requires 256 hashes to be computed on every leaf modification. A second optimization replaces subtrees consisting of exactly one leaf with a single node (3). In expectation, the depth of any given item is O(log n), where n is the number of items in the tree. This optimization reduces the number of hashes to be computed when performing operations on the map.
2^256大小的树是难以处理的表示，可以应用优化来形成稀疏的 Merkle 树。 图5显示了两个优化，这些优化用于将一个幼稚的实现(1)转化为一个高效的实现。 首先，完全由空节点组成的子树被替换为占位符值(2)，就像证书透明度系统中使用的那样[27]。 这种优化创建了可伸缩大小的表示，而不会大幅改变 Merkle 树的证明生成。 然而，树叶总是存储在树的底层，这意味着该结构需要对每一个树叶修改计算256个哈希。 第二次优化用单个节点(3)取代了完全由一个叶子组成的子树。 在期望中，任何给定项的深度都是 O (log n)，其中 n 是树中的项数。 这种优化减少了在地图上执行操作时需要计算的哈希数量。

Efficient implementations, such as Libra Core, can optimize their disk layout and batch layers of nodes to avoid seeks during lookup.
高效的实现，如 Libra Core ，可以优化它们的磁盘布局和批处理层。在查找过程中寻求避免节点。

When the sparse Merkle tree is updated after the execution of a transaction, the new tree reuses unchanged portions of the previous version, forming a persistent data structure [28, 29]. If a transac- tion modifies m accounts out of n accounts in the system, on average O(m · log n) new branches and leaves are created in the new ledger state tree that differ from the previous version.

This approach allows validators to store multiple versions of the ledger state efficiently. This feature also allows the efficient recomputation of the ledger state authenticator after processing a transaction.
当交易执行后更新稀疏的 Merkle 树时，新树会重复使用上一个版本的不变部分，形成一个持久的数据结构[28，29]。如果一个交易在系统中修改了 n 个帐户中的 m 帐户，平均来说，新分类账状态树中创建了 O (m · log n)新的分支和叶子，这些分支和叶子不同于上一个版本。这种方法允许验证器高效地存储多个版本的分类账状态。该特性还允许在处理一个事务后高效地重新计算分类账状态验证器。

We considered AVL-based trees, which provide more optimal worst-case proof length than sparse Merkle trees [30]. However, the sparse Merkle tree approach allows implementers to make optimiza- tions, such as sharding storage across servers and parallelizing root hash computation.
我们考虑了基于 AVL 的树，它提供了比稀疏 Merkle 树更优化的最坏情况证明长度[30]。然而，稀疏 Merkle 树方法允许实现者进行优化，例如跨服务器的共享存储和并行根哈希计算。

4.4. Accounts

At the logical level, an account is a collection of resources and modules stored under the account address. At the physical level, an account is treated as an ordered map of access paths to byte array values. An access path is a delimited string similar to a path in a file system.
在逻辑层面，账户是存储在账户地址下的资源和模块的集合；在物理层面，账户被视为访问路径对字节数组值的有序映射；访问路径是与文件系统中路径相似的分界字符串。

In the first iteration of the protocol, we serialize an account as a list of access paths and values sorted by access path. The authenticator of an account is the hash of this serialized representation. Note that this representation requires recomputing the authenticator over the full account after any modification to the account. The cost of this operation is O(n), where n is the length of the byte representation of the full account. Furthermore, reads from clients require the full account information to authenticate any specific value within it.
在协议的第一次迭代中，我们将一个帐户序列化为访问路径和按访问路径排序的值列表。一个帐户的验证器是这个序列化表示的哈希。请注意，这个表示需要在对帐户进行任何修改后，重新计算整个帐户的验证器。这个操作的成本是 O (n)，其中 n 是整个帐户的字节表示的长度。此外，从客户端读取需要完整帐户信息来验证它中的任何特定值。

Our strategy for storing data within accounts differs from other systems such as Ethereum. Our approach allows the Move language to provide better programming abstractions by representing accounts as ordered maps of access paths to values. This representation allows Move to efficiently manage the conservation of resources through the VM. Move encourages each user to hold

resources in their own account. In the initial release of Libra, we optimize for small accounts. In future releases, we may consider supporting more efficient structures to represent larger accounts, such as Merkle AVL trees [30].
我们在账户内存储数据的策略与以太坊等其他系统不同。我们的方法允许Move语言提供更好的编程抽象，通过将账户表示为访问路径的有序映射到值。这种表示允许移动通过 VM 高效管理资源的守恒。Move鼓励每个用户将资源持有在自己的账户中。在 Libra 的初始版本中，我们针对小账户进行优化。在未来的版本中，我们可能会考虑支持更高效的结构来表示更大的账户，例如 Merkle AVL 树[30]。

Account Eviction and Recaching. We anticipate that as the system is used, eventually storage growth associated with accounts may become a problem. Just as gas encourages responsible use of computation resources (see Section 3.1), we expect that a similar rent-based mechanism may be needed for storage. We are assessing a wide range of approaches for a rent-based mechanism that best suits the ecosystem. We discuss one option that can be applied to any policy that determines an expiration time after which data can be evicted.
账户删除和回收。我们预计，随着系统的使用，最终与账户相关的存储增长可能会成为一个问题。正如天然气鼓励负责任地使用计算资源(见第3.1节)，我们预计可能需要一个类似的基于租金的机制来存储。我们正在评估最适合生态系统的基于租金的机制的广泛方法。我们讨论了一个选项，可以适用于任何决定数据删除到期时间的策略。

After each transaction execution affecting an account, the VM computes the logical expiration time at which the storage allocated for an account can be freed. The VM is free to apply any deterministic approach to determine the expiration time. One example of such a policy would be to charge a fee denominated in Libra coins that is based on the amount of time the account is stored and its size. The account’ s authenticator is represented as H(H(AccountBlob) ∥ ExpirationTime), which encodes the expiration time for the account. After expiration, the VM denies access to the account, raising an error. Since the AccountBlob is inaccessible, a validator is free to prune this data after ExpirationTime. However, a validator allows a transaction to reactivate the account by recaching its contents after the expiration time. The transaction must contain the preimage of the AccountBlob and have an associated transaction fee that covers the cost of further storage. The authenticity of the recached contents is ensured through recomputing and checking the hash value associated with the account, which is not deleted. This method to implement rent is an improvement on existing account eviction schemes, which require a third party to send a transaction to delete the storage of the account.
每次交易执行影响账户后， VM 计算分配给账户的存储可以释放的逻辑到期时间。 VM 可以自由地应用任何确定性方法来确定到期时间。 这种政策的一个例子是收取Libra硬币计价的费用，该费用是根据账户存储的时间和规模计算的。 账户的认证人表示为 H (H (Account Blob) || ExpirationTime)，它编码了账户的到期时间。 到期后，虚拟机拒绝访问账户，导致错误。 由于无法访问到“会计器”，验证器可

以自由地在“过期时间”后对该数据进行修剪。 然而，验证器允许交易在到期后重新激活账户内容。 交易必须包含注册帐户的预图像，并有相关的交易费用，以支付进一步存储的费用。 通过重计算和检查账户关联的哈希值，确保重写内容的真实性，该哈希值不被删除。 这种实现租金的方法是对现有账户驱逐方案的改进，这些方案需要第三方发送交易来删除账户的存储。

4.5. Events

E_i is the list of events emitted during the execution of T_i. Each event is stored as a leaf in a Merkle tree that forms an authenticator for E_i. An event is serialized as a tuple of the form (A,p,c), which represents the access path of the event structure that emitted the event (A), the data payload (p), and the counter value (c). These tuples are indexed by the order j in which the events were emitted during the execution of T_i. By convention, we denote an event $j \rightarrow (A,p,c)$ being included in E_i as $(j, (A, p, c)) \in E\_i$. The authenticator for E_i is included in TransactionInfo_i . Thus, a validator can construct an inclusion proof that within the ith transaction the jth event was (A,p,c).
E_i 是执行 T_i 时发出的交易事件。 每个事件都作为叶子存储在 Merkle 树中，该树构成了 E_i 的验证器。 事件序列化为表单(A,p,c)的元组，表示发射事件(A)、数据有效载荷(p)和计数值(c)的事件结构的访问路径。 这些元组由在执行 T_i 时产生事件的 j 顺序索引。 通过惯例，我们表示事件 $j \rightarrow (A,p,c)$ 被包含在 E_i 中的为 $(j, (A, p, c)) \in E\_i$。 E_i 的验证器包含在 TransactionInfo_i 中。 因此，验证器可以构造一个包含证明，在第i笔交易中第j个事件是(A,p,c)。

The counter c, included in each event, plays a special role in allowing clients to retrieve a list of events for a given access path A. Clients can also be assured that the list is complete. An event structure representing events with access path A within the Move language maintains a counter C for the total number of events it has emitted. This counter is stored within the ledger state and incremented after each transaction execution emits an event on access path A.
每个事件中包含的计数器 c 在允许客户端检索给定访问路径的事件列表方面发挥了特殊的作用。客户端也可以放心该列表是完整的。在Move语言中代表具有访问路径 A 的事件结构维护了一个计数器 C ，该计数器存储在分类账状态中，并在每次交易执行发送访问路径 A 上的事件增量。

A client can obtain an authenticator for a recent ledger state, query the counter associated with the event structure for access path A, and retrieve the total number of events C. Then the client can query an untrusted service for the list of events on access path A. The query response consists of a set of tuples (i, j, A, p, c) — one corresponding to each event — where i is the transaction sequence number emitting the event and j is the order of the event emission within this transaction. Associated proofs of inclusion for $(j, (A, p, c)) \in Ei$ can be provided for each event. Since the client knows the total number of events emitted by access path A, they can ensure that

the untrusted service has provided this number of distinct events and also order them by their sequence number 0 ⩽ c < C. This convinces the client that the list of events returned for A is complete.
客户端可以获取最近分类账状态的验证器，查询访问路径 A 的事件结构关联的计数器，检索事件总数 C 。 然后客户端可以为访问路径 A 上的事件列表查询一个不可信任的服务。 查询响应由一组元组(i, j, A, p, c) 组成-每个事件对应的元组-其中 i 是发生事件的交易序列号， j 是该交易中事件发生的顺序。 可以为每个事件提供 (j, (A, p, c)) ∈ Ei的相关包含证明。 由于客户端知道访问路径 A 发出的事件总数，他们可以确保不受信任的服务已经提供了该数量的不同事件，并按其序列号0⩽ c < C 对它们进行排序。 这使客户确信为 A 返回的事件列表是完整的。

This scheme allows the client to hold an authenticated subscription to events on access path A. The client can periodically poll the total counter C for event access path A to determine if the subscription is up-to-date. For example, a client can use this to maintain a subscription to incoming payment transactions on an address it is watching. Untrusted third-party services can provide feeds for events indexed by access path, and clients can efficiently verify the results they return.
该方案允许客户端持有对访问路径上事件的身份验证订阅。客户端可以周期性地对事件访问路径 A 的总计数器 C 进行轮询，以确定订阅是否是最新的。例如，客户端可以使用此来维护对正在观看的地址上的进来支付交易的订阅。不可信任的第三方服务可以为访问路径索引的事件提供提要，客户端可以高效地验证他们返回的结果。

5. Byzantine Fault Tolerant Consensus

The consensus protocol allows a set of validators to create the logical appearance of a single database. The consensus protocol replicates submitted transactions among the validators, executes potential transactions against the current database, and then agrees on a binding commitment to the ordering of transactions and resulting execution. As a result, all validators can maintain an identical database for a given version number following the state machine replication paradigm [31]. The Libra Blockchain uses a variant of the HotStuff [13] consensus protocol called LibraBFT. It provides safety and liveness in the partial synchrony model in the tradition of DLS [32] and PBFT [33] as well as the newer Casper [34] and Tendermint [35]. This section outlines the key decisions in LibraBFT. A more detailed analysis, including proofs of safety and liveness, is covered in the full report on LibraBFT [12].
共识协议允许一组验证器创建单个数据库的逻辑外观。 共识协议在验证者之间复制提交的交易，针对当前数据库执行潜在的交易，然后就交易的顺序和由此产生的执行达成有约束力的承诺。 因此，所有验证器都可以按照状态机复制范式[31]，为给定的版本号维护一个相同的数据库。 Libra Blockchain 使用

Hot Stuff [13]共识协议的变体，称为 LibraBFT 。 它在传统的 DLS [32]和 PBFT [33]以及较新的 Casper [34]和 Tendermint [35]中提供了部分同步模型的安全性和活动性。 本节概述了 LibraBFT 中的关键决策。 更详细的分析，包括安全性和活动性的证明，见关于 LibraBFT 的完整报告[12]。

Agreement on the database state must be reached between validators, even if there are Byzantine faults [36]. The Byzantine failures model allows some validators to arbitrarily deviate from the protocol without constraint, except for being computationally bounded (and thus not able to break cryptographic assumptions). Byzantine faults are worst-case errors where validators collude and behave maliciously to try to sabotage system behavior. A consensus protocol that tolerates Byzantine faults caused by malicious or hacked validators can also mitigate arbitrary hardware and software failures.
验证者之间必须达成关于数据库状态的协议，即使存在拜占庭故障[36]。拜占庭故障模型允许一些验证者在没有约束的情况下任意偏离协议，除了被计算界（从而无法打破密码学假设）。拜占庭故障是最糟糕的错误，验证者串通和恶意行为试图破坏系统行为。一个容忍由恶意或被黑客攻击的验证者造成的拜占庭故障的共识协议也可以减轻任意的硬件和软件故障。

LibraBFT assumes that a set of 3f + 1 votes is distributed among a set of validators that may be honest, or Byzantine. LibraBFT remains safe, preventing attacks such as double spends and forks when at most f votes are controlled by Byzantine validators. LibraBFT remains live — committing transactions from clients — as long as there exists a global stabilization time (GST), after which all messages between honest validators are delivered to other honest validators within a maximal network delay δ (this is the partial synchrony model introduced in [32]). In addition to traditional guarantees, LibraBFT maintains safety when validators crash and restart — even if all validators restart at the same time.
LibraBFT 假设一组3f +1票分布在一组验证者中，这些验证者可能是诚实的，也可能是拜占庭的。 LibraBFT 仍然是安全的，可以防止双花和分叉等攻击，最多 f 票由拜占庭验证者控制。 只要存在一个全局稳定时间(GST)，在该时间之后，诚实验证器之间的所有消息都在最大网络延迟δ内交付给其他诚实验证器(这是[32]中引入的部分同步模型)， LibraBFT 仍然是活的-从客户端提交交易。 除了传统的保证， LibraBFT 在验证器崩溃和重启时保持安全——即使所有验证器同时重启。

Overview of LibraBFT. Validators receive transactions from clients and share them with each other through a shared mempool protocol. The LibraBFT protocol then proceeds in a sequence of rounds. In each round, a validator takes the role of leader and proposes a block of transactions to extend a certified sequence of blocks (see quorum certificates below) that contain the full previous transaction history. A validator receives the proposed block and checks their voting rules to determine if it should vote for certifying this block. These simple rules ensure the safety of LibraBFT — and their implementation can be cleanly separated and audited.10 If the validator

intends to vote for this block, it executes the block’ s transactions speculatively and without external effect. This results in the computation of an authenticator for the database that results from the execution of the block. The validator then sends a signed vote for the block and the database authenticator to the leader. The leader gathers these votes to form a quorum certificate (QC), which provides evidence of ⩾ 2f + 1 votes for this block and broadcasts the QC to all validators.
LibraBFT 概述。 验证者从客户端接收交易，并通过共享备忘录池协议相互共享。 然后， LibraBFT 协议进行了一系列的轮次。 在每个回合中，验证器扮演领导者的角色，并提出一个交易块，以扩展包含完整先前交易历史的经过认证的块序列（见下文法定人数证书）。 验证器接收提议的区块并检查它们的投票规则，以确定它是否应该投票认证该区块。 这些简单的规则确保了 LibraBFT 的安全--它们的实现可以被干净地分离和审计。如果验证者打算投票给这个区块，它会以投机的方式执行区块的交易，而不会产生外部影响。 这导致了对执行块的数据库的验证器的计算。 然后验证器将对块和数据库验证器的签名投票发送给领导者。 领导者收集这些选票形成法定人数证书(QC)，该证书为该区块提供⩾2f +1票的证据，并向所有验证者广播 QC 。

A block is committed when a contiguous 3-chain commit rule is met. A block at round k is committed if it has a QC and is confirmed by two more blocks and QCs at rounds k+1 and k+2. The commit rule eventually allows honest validators to commit a block. LibraBFT guarantees that all honest validators will eventually commit the block (and proceeding sequence of blocks linked from it). Once a sequence of blocks has committed, the state that results from executing their transactions can be persisted and forms a replicated database.
当满足一个连续的3链提交规则时，一个区块就被提交了。如果 k 轮的区块有一个 QC ，并且在 k +1和 k +2轮得到两个以上的区块和 QC 的确认，就被提交了。提交规则最终允许诚实验证者提交一个区块。 LibraBFT 保证所有诚实验证者最终都将提交该区块(以及从该区块中链接的区块序列)。一旦一系列区块被提交，执行其交易产生的状态可以被持续并形成一个复制的数据库。

Advantages of the HotStuff paradigm. We evaluated several BFT-based protocols against the dimensions of performance, reliability, security, ease of robust implementation, and operational over- head for validators. Our goal was to choose a protocol that would initially support at least 100 validators and would be able to evolve over time to support 500–1,000 validators. We had three reasons for selecting the HotStuff protocol as the basis for LibraBFT: (1) simplicity and modularity of the safety argument; (2) ability to easily integrate consensus with execution; and (3) promising performance in early experiments.
Hot Stuff 范式的优点. 我们 比较了若干基于 BFT 的协议，对照性能、可靠性、安全性、鲁棒实现的易用性和操作性等方面对验证器进行了评估 .我们的目标是选择一个协议，该协议最初支持至少100个验证器，并且随着时间的推移能够进化到支持500-1000个验证器.我们有三个理由选择 Hot Stuff 协议作为

LibraBFT 的基础:(1)安全论点的简单性和模块化；(2)能够轻松地将共识与执行集成；(3)在早期实验中有希望的性能。

The HotStuff protocol decomposes into modules for safety (voting and commit rules) and liveness (pacemaker). This decoupling provides the ability to develop and experiment independently and on different modules in parallel. Due to the simple voting and commit rules, protocol safety is easy to implement and verify. It is straightforward to integrate execution as a part of consensus to avoid forking issues that arise from non-deterministic execution in a leader-based protocol. Finally, our early prototypes confirmed high throughput and low transaction latency as independently measured in HotStuff [13]. We did not consider proof-of-work based protocols due to their poor performance
and high energy (and environmental) costs [24].
Hot Stuff 协议分解为安全（投票和提交规则）和活力（起搏器）的模块。这种解耦提供了独立开发和实验的能力，并并行在不同的模块上。由于投票和提交规则的简单，协议安全易于实现和验证。将执行集成为共识的一部分是很简单的，以避免基于领导者的协议中非确定性执行产生的伪造问题。最后，我们的早期原型确认了在 Hot Stuff [13]中独立测量的高吞吐量和低交易延迟。由于它们性能不佳，我们没有考虑基于工作证明的协议。以及高昂的能源(和环境)成本[24]。

HotStuff extensions and modifications. In LibraBFT, to better support the goals of the Libra ecosystem, we extend and adapt the core HotStuff protocol and implementation in several ways. Importantly, we reformulate the safety conditions and provide extended proofs of safety, liveness, and optimistic responsiveness. We also implement a number of additional features. First, we make the protocol more resistant to non-determinism bugs by having validators collectively sign the resulting state of a block rather than just the sequence of transactions. This also allows clients to use QCs to authenticate reads from the database. Second, we design a pacemaker that emits explicit timeouts, and validators rely on a quorum of those to move to the next round — without requiring synchronized clocks. Third, we design an unpredictable leader election mechanism in which the leader of a round is determined by the proposer of the latest committed block using a verifiable random function [37]. This mechanism limits the window of time in which an adversary can launch an effective denial-of- service attack against a leader. Fourth, we use aggregate signatures [38] that preserve the identity validators who sign QCs. This allows us to provide incentives to validators that contribute to QCs
(discussed in Section 9.3). It also does not require a complex threshold key setup [39].
Hot Stuff 扩展和修改。 在 LibraBFT 中，为了更好地支持 Libra 生态系统的目标，我们以几种方式扩展和调整了核心 HotStuff 协议和实现。 重要的是，我们重新制定了安全条件，并提供了安全、活泼和乐观反应的扩展证明。 我们还实现了一些额外的功能。 首先，我们让验证器集体签署块的结果状态，而不仅仅是事务序列，从而使协议更能抵御非决定论错误。 这也允许客户端使用 QCs 来验证数据库中的

读取。 其次，我们设计了一个发出明确超时的起搏器，验证器依赖于进入下一轮的法定人数——而不需要同步时钟。 第三，我们设计了一个不可预测的领导者选举机制，其中一轮的领导者由最新提交块的提议者使用可验证的随机函数确定[37]。 该机制限制了对手对领导者发起有效的拒绝服务攻击的时间窗口。 第四，我们使用聚合签名[38]来保存签署 QCs 的身份验证者。 这使我们能够向为 QC 做出贡献的验证者提供激励(在第9.3节中讨论)。它也不需要复杂的阈值密钥设置[39]。

Validator management. The Libra protocol manages the set of validators using a Move module. This creates a clean separation between the consensus system and the cryptoeconomic system that defines a trusted set of validators. We discuss the Libra Blockchain’ s implementation of this contract in Section 9.2. Abstracting the validator management in this way is an example of the flexibility granted by defining core blockchain primitives in Move.
验证器管理。 Libra 协议使用Move模块管理验证器集。这在共识系统和定义可信验证器集的密码经济系统之间创建了一个干净的分离。我们在第9.2节讨论了 Libra Blockchain 对该合同的实施。以这种方式抽象验证器管理是通过在Move中定义核心区块链基元赋予的灵活性的一个例子。

Each change to the group of validators defines a new epoch. If a transaction causes the validator management module to alter the validator set, that transaction will be the last transaction committed by the current epoch — any subsequent transactions in that block or future blocks from that epoch will be ignored. Once the transaction has been committed, the new set of validators can start the next epoch of the consensus protocol.
对验证器组的每一次更改都定义了一个新的时代。如果一个交易导致验证器管理模块改变验证器集，该交易将是当前周期提交的最后一个交易-该块或该周期的未来块中的任何后续交易将被忽略。一旦提交了交易，新的验证器集可以开始共识协议的下一个时代。

Within an epoch, clients do not need to synchronize every QC. Since a committed QC contains a binding commitment to all previous states, a client only needs to synchronize to the latest available QC in its current epoch. If this QC is the last in its epoch, the client can see the new set of validators, update its epoch, and again synchronize to the latest QC. If a validator chooses to prune history as described in Section 4.2, it needs to retain at least enough data to provide proof of the validator set change to clients.
在一个时代内，客户端不需要同步每个 QC 。由于承诺的 QC 包含对所有先前状态的绑定承诺，客户端只需要同步到当前时代的最新可用 QC 。如果这个 QC 是其时代中的最后一个，客户端可以看到新的验证器集，更新其时代，并再次同步到最新的 QC 。如果验证器选择剪切历史，如第4.2节所述，它需要保留至少足够的数据，以向客户端提供验证器集变化的证明。

The validator management contract must provide a validator set that satisfies the security properties required by the consensus protocol. No more than f voting power can be controlled by

Byzantine validators. The voting power must remain honest both during the epoch as well as for a period time after the epoch in order to allow clients to synchronize to the new configuration. A client that is offline for longer than this period needs to resynchronize using some external source of truth to acquire a checkpoint that they trust (e.g., from the source it uses to receive updated software). Furthermore, the validator set must not rotate so frequently that the rotation disrupts the performance of the system or causes clients to download QCs for an excessive number of epochs. We plan to research the optimal epoch length but anticipate it to be less than a day.
验证器管理合同必须提供一个满足共识协议要求的安全属性的验证器集。 不超过 f 的投票权可以由拜占庭验证者控制。 投票权必须保持诚实，既在时代期间，也在时代之后的一段时间内，以允许客户同步到新配置。 一个离线时间超过这个周期的客户端需要使用一些外部的真相来源重新同步以获得他们信任的检查点（例如，从它用来接收更新软件的来源）。 此外，验证器集不得如此频繁地旋转，以至于旋转扰乱系统的性能或导致客户端下载数量过多的 QC 。 我们计划研究最佳时代长度，但预计不到一天。

6. Networking

The Libra protocol, like other decentralized systems, needs a networking substrate to enable commu- nication between its members. Both the consensus and shared mempool protocols between validators require communication over the internet, as described in Section 5 and Section 7, respectively.
Libra 协议和其他去中心化系统一样，需要一个网络基础来实现成员之间的通信。验证者之间的共识协议和共享备忘录池协议都需要通过互联网进行通信，分别见第5节和第7节。

The network layer is designed to be general-purpose and draws inspiration from the libp2p [40] project. It currently provides two primary interfaces: (1) Remote Procedure Calls (RPC) and (2) DirectSend, which implements fire-and-forget-style message delivery to a single receiver.
该网络层设计为通用型，并从 libp2p [40]项目中汲取灵感，目前提供了两个主要接口：(1)远程程序调用(RPC)和(2) DirectSend，该接口实现了对单个接收器的 fire-and-forget-style 消息传递。

The inter-validator network is implemented as a peer-to-peer system using Multiaddr [41] scheme for peer addressing, TCP for reliable transport, Noise [42] for authentication and full end-to-end encryption, Yamux [43] for multiplexing substreams over a single connection, and push-style gossip for peer discovery. Each new substream is assigned a protocol supported by both the sender and the receiver. Each RPC and DirectSend type corresponds to one such protocol.
跨验证器网络是作为一个点对点系统来实现的，使用 Multiaddr [41]方案进行点对点寻址， TCP 用于可靠传输， Noise [42]用于认证和完全端到端加密， Yamux [43]用于在单个连接上复用子流，以及用于点

发现的推送式八卦。每个新的子流都被分配一个由发送方和接收方支持的协议。每个 RPC 和 DirectSend 类型对应一个这样的协议。

The networking system uses the same validator set management smart contract as the consensus system as a source of truth for the current validator set. This contract holds the network public key and consensus public key of each validator. A validator detects changes in the validator set by watching for changes in this smart contract. To join the inter-validator network, a validator must authenticate using a network public key in the most recent validator set defined by the contract. Bootstrapping a validator requires a list of seed peers, which first authenticate the joining validator as an eligible member of the inter-validator network and then share their state with the new peer.
网络系统使用与共识系统相同的验证器集管理智能合约作为当前验证器集的真理来源。 本合同持有每个验证器的网络公钥和共识公钥。 验证器通过观察此智能合约中的变化来检测验证器集中的变化。 要加入跨验证器网络，验证器必须使用合同定义的最近验证器集中的网络公钥进行验证。 启动验证器需要一个种子同行列表，该列表首先将加入验证器认证为跨验证器网络的合格成员，然后将它们的状态与新的同行共享。

Each validator in the Libra protocol maintains a full membership view of the system and connects directly to any validator it needs to communicate with. A validator that cannot be connected to directly is assumed to fall within the quota of Byzantine faults tolerated by the system. Validator health information, determined using periodic liveness probes, is not shared between validators; in- stead, each validator directly monitors its peers for liveness. We expect this approach to scale up to a few hundred validators before requiring partial membership views, sophisticated failure detectors, or communication relays.
Libra 协议中的每个验证器维护系统的完整成员视图，并直接连接到它需要通信的任何验证器。不能直接连接到的验证器被假设属于系统容忍的拜占庭故障的配额范围内。验证器健康信息，使用周期灵敏度探针确定，在验证器之间不共享；取而代之的是，每个验证器直接监控其同行的灵敏度。我们预计这种方法将扩大到几百个验证器，然后需要部分成员视图、复杂的故障检测器或通信继电器。

7. Libra Core Implementation

To validate the Libra protocol, we have built an open-source prototype implementation, Libra Core [6]. The implementation is written in Rust. We chose Rust due to its focus on enabling safe coding practices, support for systems programming, and high performance. We have split the internal components of the system as gRPC [44] services. Modularity allows for better security; for

instance, the consensus safety component can be run in a separate process or even on a different machine.
为了验证 Libra 协议，我们构建了一个开源的原型实现， Libra Core [6]。实现是用 Rust 编写的。我们选择 Rust 是因为它专注于启用安全的编码实践、支持系统编程和高性能。我们已经将系统的内部组件拆分为 gRPC [44]服务。模块化允许更好的安全性；例如，共识安全组件可以在单独的进程中运行，甚至在不同的机器上运行。

The security of the Libra Blockchain rests on the correct implementation of validators, Move programs, and the Move VM. Addressing these issues in Libra Core is a work in progress. It involves isolating the parts of the code that contribute to a validator signing a block of transactions during consensus and applying measures to increase assurance in the correctness of these components (e.g., extensive testing, formal specification, and formal verification). Developing high assurance also involves ensuring the security of the dependencies of the code (e.g., the code review process, source control, open-source library dependencies, build systems, and release management).
Libra 区块链的安全性取决于验证器、Move程序和Move虚拟机的正确实现。在 Libra Core 中解决这些问题是一项正在进行的工作。它涉及隔离代码中有助于验证器在共识期间签署交易块的部分，并应用措施来增加这些组件的正确性(例如，广泛测试、正式规范和正式验证)。开发高度保证还包括确保代码依赖关系的安全性(例如，代码审查过程、源代码控制、开源库依赖关系、构建系统和发布管理)。

7.1. Write Request Lifecycle

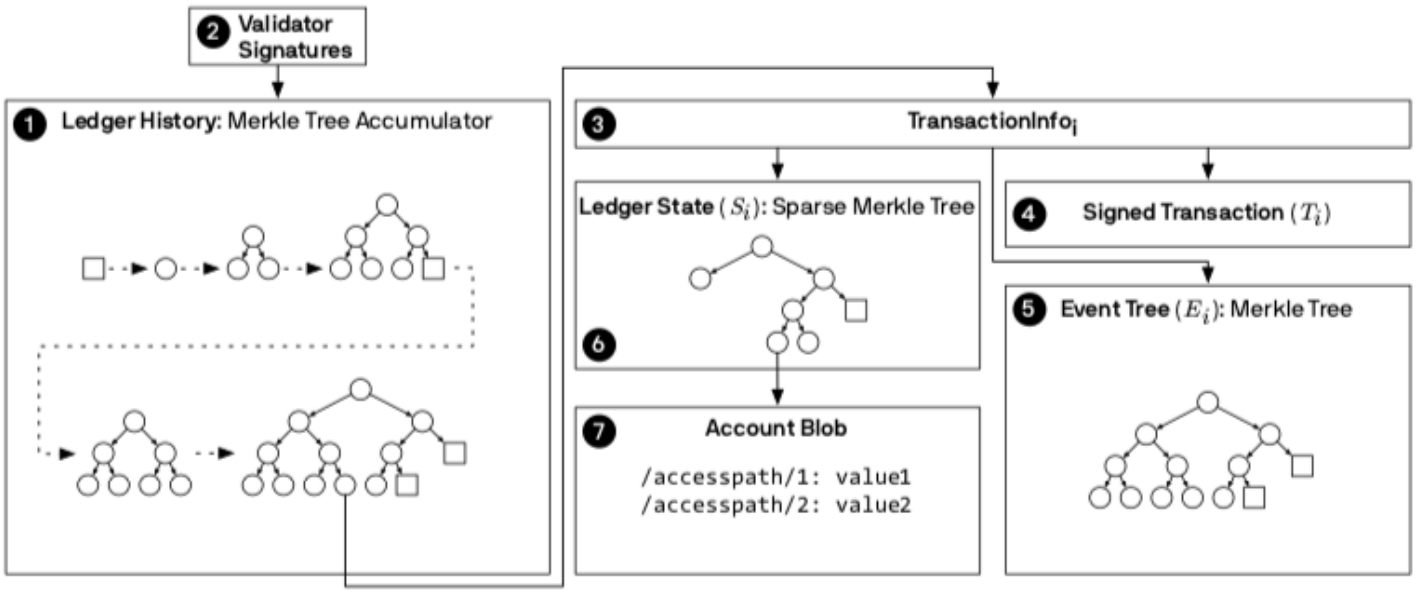

Figure 3: (❶) The root hash of the ledger history structure is the authenticator to the full state of the system that is (❷) signed by a quorum of validators. As transactions are added to the database, the authenticator (Section 4.2) committing to the ledger history grows (depicted with dashed arrows). Each leaf of the ledger history commits to a (❸) TransactionInfo structure. This structure commits to the (❹) signed transaction (T_i), (❺) the list of events generated during that transaction (E_i, Section 4.5), and the (❻) state after the execution of that transaction (S_i, Section 4.3). The state is a sparse Merkle tree with an (❼) account blob at each leaf.

Figure 6 shows the lifecycle of transactions within Libra Core that support write operations to the decentralized database. This section takes an in-depth look at how a transaction flows through the internal components of the validators.
图6显示了 Libra Core 中支持向去中心化数据库写入操作的交易的生命周期。本节深入研究交易如何通过验证器的内部组件流转。

A request begins when a client wishes to submit a transaction to the database. We currently assume
clients have an out-of-band mechanism to find the addresses of validators to submit transactions to — the final version of this mechanism is yet to be designed.
当客户端希望向数据库提交交易时，请求开始。客户端有一个带外机制，可以找到验证器的地址，以便向客户端提交交易--该机制的最终版本尚未设计完成。

Admission control. Upon receiving a transaction, a validator’ s admission control component per- forms initial syntactic checks (1) to discard malformed transactions that will never be executed. Doing these checks as early as possible avoids spending resources on transactions that spam the system. Admission control may access the VM (2), which uses the storage component to perform checks, such as ensuring the account has a sufficient balance to pay the gas for the

transaction. The admission control component is designed so that a validator can run multiple instances of the compo- nent. This design allows the validator to scale the processing of incoming transactions and mitigate denial-of-service attacks.
入场控制。 在收到交易时，验证器的入场控制组件通过初始句法检查(1)来丢弃永远不会执行的错误交易。 尽早进行这些检查可以避免将资源用于垃圾邮件系统的交易。 入场控制可以访问虚拟机(2)，虚拟机使用存储组件执行检查，例如确保账户有足够的余额来支付交易的气体。 输入控制组件的设计是为了使验证器能够运行该组合的多个实例。 这种设计允许验证器扩展对传入交易的处理并减轻拒绝服务攻击。

Mempool. Transactions that pass the checks of the admission control component are sent to the validator’ s mempool, which holds transactions waiting to be executed in an in-memory buffer (3). The mempool may hold multiple transactions sent from the same address. Because the mempool processes multiple transactions at a time, it can perform checks such as validating that a sequence of operations on the same address can all pay for gas that the admission control system cannot. Using the shared-mempool protocol (4), a validator shares the transactions in its mempool with other validators and places transactions received from other validators in its own mempool.
内存池。 通过入场控制组件检查的交易将被发送到验证器的内存池，该内存池持有等待在内存缓冲区(3)中执行的交易。 内存池可以保存从同一地址发送的多个交易。 因为内存池一次处理多个交易，它可以执行检查，例如验证同一地址上的一系列操作可以支付入场控制系统无法支付的气体。 使用共享内存池协议(4)，验证器与其他验证器共享备内存吃中的交易，并将从其他验证器收到的交易放在自己的内存池中。

Consensus. Validators create blocks by selecting a sequence of transactions from their mempool. When a validator acts as the leader of the consensus protocol, it forms a block of transactions from its mempool (5). It sends this block of transactions as a proposal to other validators (6). The consensus component is responsible for coordinating agreement among all validators on the sequence of these blocks of transactions and their resulting execution using the LibraBFT protocol (Section 5).
共识。验证者通过从他们的内存池中选择一个交易序列来创建区块。当验证者作为共识协议的领导者时，它从其备忘录池(5)中形成一个交易区块。它将这个交易区块作为提议发送给其他验证者(6)。共识组件负责协调所有验证者之间关于这些交易区块序列的协议及其使用 LibraBFT 协议的执行(第5节)。

Transaction execution. As part of reaching agreement, the block of transactions is passed to the ex- ecutor component (7), which manages the execution of transactions (Section 3) in the VM (8). This execution happens speculatively before the transaction has been agreed on. This early execution is safe since transactions are deterministic and have no external effects. After executing

the transactions in the block, the execution component builds a ledger history (Section 4.2) with these transactions appended. The ledger history authenticator is returned to the consensus component (9).
交易执行。作为达成协议的一部分，交易块被传递给执行部分(7)，执行部分在虚拟机(8)中管理交易的执行(第3节)。这种执行发生在交易达成一致之前。这种早期执行是安全的，因为交易是确定性的，没有外部影响。在执行块中的交易后，执行部分在附加这些交易的情况下建立一个分类账历史(第4.2节)。分类账历史验证器被返回到共识部分(9)。

The leader then attempts to reach consensus on this authenticator by forming a chain of quorum certificates (Section 5), each of which is signed by a set of validators with at least 2f + 1 votes.
然后，领导者试图通过形成一系列法定人数证书(第5节)来达成对该认证人的共识，每一证书都由一组至少有2f +1票的验证人签署。

Committing a block. Once the consensus algorithm reaches agreement, any honest validator can be sure that all other honest validators will eventually commit a consistent ledger history. The validator reads the result of the block execution from the cache in the execution component and updates its local database storage (10).
提交块。一旦共识算法达成一致，任何诚实验证器都可以确定所有其他诚实验证器最终都会提交一致的分类账历史。验证器从执行组件中的缓存读取块执行的结果，并更新其本地数据库存储(10)。

7.2. Read Request Lifecycle

Clients may also submit read requests to query validators for the content of an account in the decentralized database. Read requests do not mutate state and can be processed locally without going through consensus. Reads are submitted to the validator’ s admission control component. The admission control component performs preliminary checks, reads the data from storage, and the result is sent back to the client. The response comes with a quorum certificate (described in Section 5) that contains a root hash (described in Section 4). The QC allows the client to authenticate the response to the query. The client uses the same technique as the VM to interpret the raw bytes of an account using the logical data model (Section 2). For example, to read the balance of the account at address a, the client decodes the LibraCoin.T resource embedded inside the account’ s raw bytes.
客户端也可以向查询验证器提交关于分散数据库中帐户内容的读取请求。 读取请求不会发生状态突变，可以在不经过共识的情况下进行本地处理。 读数提交给验证器的入场控制组件。 收录控制组件执行初步检查，从存储中读取数据，并将结果发回客户端。 响应附带一个包含根散列（在第4节中描述）的法定人数证书。 QC 允许客户机对查询的响应进行身份验证。 客户端使用与 VM 相同的技术，使用逻辑数

据模型解释账户的原始字节（第2节）。 例如，要读取地址 a 的账户余额，客户端解码 LibraCoin.T 资源嵌入到帐户的原始字节中。

8. Performance

The mission of the Libra protocol is to support a global financial infrastructure. Performance is an integral part of meeting that need. We discuss three components of blockchain performance:
Libra 协议的使命是支持全球金融基础设施。性能是满足这一需求的一个组成部分。我们讨论了区块链性能的三个组成部分:

1. **Throughput**: The number of transactions that the blockchain can process per second.
1. 吞吐量：区块链每秒可以处理的交易数量。

2. **Latency**: The time between a client submitting a transaction to the blockchain and another party seeing that the transaction was committed.
2. 延迟:客户端向区块链提交交易和另一方看到交易被提交之间的时间。

3. **Capacity**: The ability of the blockchain to store a large number of accounts.
3. 容量：区块链存储大量账户的能力。

While the Libra protocol is still at a prototype stage and we do not have concrete performance metrics yet to report, we anticipate the initial launch of Libra protocol to support 1,000 payment transactions per second with a 10-second finality time between a transaction being submitted and committed. Over time, we expect to be able to increase the system’ s throughput to meet the needs of the network. We anticipate that many payment transactions will occur off-chain, for example, within a custodial wallet or by using payment channels [45]. Therefore, we believe that supporting 1,000 transactions per second on the blockchain will meet the initial needs of the ecosystem. The Libra protocol is designed to achieve these goals in several ways:
虽然 Libra 协议仍处于原型阶段，我们还没有具体的性能指标有待报告，但我们预计 Libra 协议将首次推出，支持每秒1000笔支付交易，交易提交和承诺之间有10秒的结束时间。 随着时间的推移，我们期望能够提高系统的吞吐量，以满足网络的需求。 我们预计，许多支付交易将发生在链外，例如，在托管钱包内或通过使用支付通道[45]。 因此，我们认为，在区块链上支持每秒1000笔交易将满足生态系统的初始需求。 Libra协议旨在以几种方式实现这些目标:

Protocol design. Many elements of the Libra protocol are chosen partly based on performance. For example, the LibraBFT algorithm achieves consensus in three rounds of network

communication and does not require any real-time delay to propose or vote on blocks. This allows the commit latency to be limited only by the network latency between validators.
协议设计。 Libra 协议的许多元素部分是基于性能来选择的。例如， LibraBFT 算法在三轮网络通信中实现共识，不需要任何实时延迟来提议或投票区块。这允许提交延迟仅受验证者之间的网络延迟限制。

We also select elements of the protocol with parallelization and sharding in mind. The sparse Merkle tree approach to computing authenticators allows sharding the database across multiple machines (which increases capacity) or processing updates in parallel (which increases throughput). Initial transaction validation, which includes computationally expensive signature verification, can also be
parallelized.
我们还在考虑并行化和共享的情况下选择协议的元素。稀疏的 Merkle 树计算认证器方法允许在多台机器上共享数据库(这增加了容量)或并行处理更新(这增加了吞吐量)。初始交易验证，包括计算上昂贵的签名验证，也可以用于并行。

Validator selection. Like most services, the performance of the Libra Blockchain depends on the performance of the underlying validators that operate it. There is a tradeoff between decentralization and performance. Requiring extremely well-resourced validators limits the number of entities that could perform that role. However, the presence of extremely under-resourced validators would limit the performance of the whole system.
验证器选择。像大多数服务一样， Libra 区块链的性能取决于操作它的底层验证器的性能。分散和性能之间有一个权衡。要求资源非常充足的验证器限制了可以执行该角色的实体的数量。然而，资源极度不足的验证器的存在将限制整个系统的性能。

We favor a balance of these approaches by targeting nodes that can run on commodity hardware that many entities can purchase. However, we do assume that nodes run on server-class hardware and within well-connected data centers. We use an approximate analysis to show that the system is likely able to meet the demand of 1,000 transactions per second.
我们支持这些方法的平衡，目标是可以运行在许多实体可以购买的商品硬件上的节点。然而，我们确实假设节点运行在服务器级硬件上，并在连接良好的数据中心内。我们使用近似分析表明，该系统很可能能够满足每秒1000笔交易的需求。

- **Bandwidth:** If we assume that each transaction requires 5 KB of traffic — including the cost of receiving the transaction via the mempool, rebroadcasting it, receiving blocks from the leader, and replicating to clients — then validators require a 40 Mbps internet connection to support 1,000 transactions per second. Access to such bandwidth is widely available.

- **带宽:** 如果我们假设每笔交易需要5 KB 的流量-包括通过内存池接收交易、重新广播交易、从领导者接收块和复制到客户端的成本-那么验证器需要40 Mbps 的互联网连接才能支持每秒1000笔交易。这种带宽的访问是广泛可用的。

- **CPU:** Signature verification is a significant computational cost associated with a payment trans- action. We have designed the protocol to allow parallel verification of transaction signatures. Modern signature schemes support over 1,000 verifications per second over a commodity CPU.
- **CPU：** 签名验证是与支付交易相关的重要计算成本。我们设计了允许并行验证交易签名的协议。现代签名方案支持超过每秒1000次的商品 CPU 验证。

- Disk: Servers with 16 TB of SSD storage are available from major server vendors. Since the current state is the only piece of information the validator needs to use to process a transaction, we estimate that if accounts are approximately 4 KB (inclusive of all forms of overhead), then this allows validators to store 4 billion accounts. We anticipate that developments in disk storage, scaling validators to multiple shards, and economic incentives will allow the system to remain accessible to commodity systems.
- **磁盘:** 具有16 TB 固态硬盘存储的服务器可以从主要服务器供应商获得。由于当前状态是验证器处理交易所需的唯一信息，我们估计，如果账户大约为4 KB (包括所有形式的开销)，那么验证器可以存储40亿个账户。我们预计磁盘存储、将验证器扩展到多个碎片和经济激励将允许系统对商品系统保持访问。

Historical data may grow beyond the amount that can be handled by an individual server. Validators are free to discard historical data not needed to process new transactions (see Section 4.2); however, this data may be of interest to clients who wish to query events from past transactions. Since the validators sign a binding commitment to this data, clients are free to use any system to access data without having to trust the system that delivers it. We expect this type of read traffic to be easy to scale through parallelism.
历史数据的增长可能超过单个服务器可以处理的数量。验证者可以自由丢弃处理新事务不需要的历史数据(见第4.2节)；然而，这些数据可能对希望从过去的事务中查询事件的客户感兴趣。由于验证者对这些数据签署了绑定承诺，客户可以自由使用任何系统访问数据，而不必相信交付数据的系统。我们预计这种类型的读取流量很容易通过并行扩展。

9. Implementing Libra Ecosystem Policies with Move

The Libra Blockchain is a unique system that balances the stability of traditional financial networks with the openness offered by systems governed by cryptoeconomic means. As we discuss in Section 1, the Libra protocol is designed to support the Libra ecosystem in implementing novel economic [4] and governance [3] policies. The protocol specifies a flexible framework that is parametric in key system components such as the native currency, validator management, and transaction validation. In this section, we discuss how the Libra Blockchain uses the Move programming language to customize these components. Our discussion focuses on both the challenges of aligning network participant and validator incentives as well as the challenges of supporting the operations, governance, and evolution of the Libra ecosystem.
Libra区块链是一个独特的系统，它平衡了传统金融网络的稳定性和由密码经济手段管理的系统提供的开放性。 正如我们在第1节中讨论的那样，Libra协议旨在支持Libra生态系统实施新的经济[4]和治理[3]政策。 该协议指定了一个灵活的框架，该框架在关键的系统组件中参数化，如本机货币、验证器管理和事务验证。 在本节中，我们讨论了 Libra Blockchain 如何使用 Move 编程语言来自定义这些组件。 我们的讨论侧重于调整网络参与者和验证者激励的挑战，以及支持天秤座生态系统的运营、治理和进化的挑战。

9.1. Libra Coin

Many cryptocurrencies are not backed by real-world assets. As a result, investment and speculation have been primary use cases. Investors often acquire these currencies under the assumption that they will substantially appreciate and can later be sold at a higher price. Fluctuations in the beliefs about the long-term value of these currencies have caused corresponding fluctuations in price, which sometimes yield massive swings in value.
许多加密货币没有现实世界资产的支持。因此，投资和投机一直是主要的用例。投资者通常在假设这些货币将大幅升值并随后可以以更高的价格出售的情况下收购这些货币。对这些货币长期价值的信念的波动引起了相应的价格波动，这有时会导致价值的巨大波动。

To drive widespread adoption, Libra is designed to be a currency where any user will know that the value of a Libra today will be close to its value tomorrow and in the future. The reserve is the key mechanism for achieving value preservation. Through the reserve, each coin is fully backed with a set of stable and liquid assets. With the presence of a competitive group of liquidity providers that interface with the reserve, users can have confidence that any coin they hold can be sold for fiat currency at a narrow spread above or below the value of the underlying assets. This gives the coin intrinsic value from the start and helps protect against the speculative swings that are experienced by existing cryptocurrencies.
为了推动广泛采用，Libra被设计成一种货币，任何用户都将知道Libra今天的价值将接近它明天和未来的价值。 储备是实现保值的关键机制。 通过储备，每个硬币都有一套稳定且流动性好的资产作为充分的支持。 有了一批有竞争力的流动性提供者的存在，他们与储备相对接，用户可以有信心，他们持有的

任何硬币都可以以高于或低于基础资产价值的窄幅价差出售菲亚特货币。 这从一开始就赋予了硬币内在价值，并有助于防止现有加密货币经历的投机性波动。

The reserve assets are a collection of low-volatility assets, including cash and government securities from stable and reputable central banks. As the value of Libra is effectively linked to a basket of fiat currencies, from the point of view of any specific currency, there will be fluctuations in the value of Libra. The makeup of the reserve is designed to mitigate the likelihood and severity of these fluctuations, particularly in the negative direction (e.g., even in economic crises). To that end, the basket has been structured with capital preservation and liquidity in mind.
储备资产是低波动性资产的集合，包括来自稳定和信誉良好的中央银行的现金和政府证券。由于Libra的价值与一篮子法定货币有效挂钩，从任何特定货币的角度来看，Libra的价值都会有波动。储备的构成旨在减轻这些波动的可能性和严重性，特别是在负面方向(例如，即使在经济危机中)。为此，篮子的结构考虑到资本保值和流动性。

The reserve is managed by the Libra Association (see Section 9.2), which has published a detailed report on the reserve’s operations [4]. Users do not directly interface with the reserve. Instead, to support higher efficiency, there are authorized resellers who are the only entities authorized by the association to transact large amounts of fiat and Libra in and out of the reserve. These authorized resellers integrate into exchanges and other institutions that buy and sell cryptocurrencies and provide these entities with liquidity for users who wish to convert from cash to Libra and back again.
储备由Libra协会管理(见第9.2节)，该协会发布了一份关于储备运营的详细报告[4]。用户不直接与储备接口。相反，为了支持更高的效率，有授权经销商，他们是该协会授权的唯一实体，可以交易大量菲亚特和Libra进出储备。这些授权经销商整合到交易所和其他买卖加密货币的机构，并为这些实体提供流动性，让希望从现金转换到天秤座并再次返回的用户。

To implement this scheme, the Libra coin contract allows the association to mint new coins when demand increases and destroy them when the demand contracts. The association does not set a monetary policy. It can only mint and burn coins in response to demand from authorized resellers. Users do not need to worry about the association introducing inflation into the system or debasing the currency: for new coins to be minted, there must be a commensurate fiat deposit in the reserve.
为了实施这一方案，天秤币契约允许协会在需求增加时铸造新币，并在需求契约时销毁新币，协会不制定货币政策，只能根据授权经销商的需求铸造和烧币，用户不需要担心协会将通货膨胀引入系统或贬低货币:要铸造新币，必须在储备中有相应的菲亚特存款。

The ability to customize the Libra coin contract using Move allows the definition of this scheme without any modifications to the underlying protocol or the software that implements it. Additional functionality can be created, such as requiring multiple signatures to mint currency and creating limited-quantity keys to increase security.
使用 Move 自定义天秤座币合约的能力允许定义这种方案，而不需要对底层协议或实现它的软件进行任何修改，可以创建额外的功能，比如需要多个签名来铸造货币，并创建限量密钥来增加安全性。

9.2 Validator Management and Governance

The consensus algorithm relies on the validator-set management Move module to maintain the current set of validators and manage the allocation of votes among the validators. This contract is responsible for maintaining a validator set in which at most f votes out of 3f + 1 total votes are controlled by Byzantine validators.
共识算法依赖于验证器集管理 Move 模块来维护当前的验证器集，并管理验证器之间的投票分配。本合同负责维护一个验证器集，其中3f +1总投票中的最多 f 票由拜占庭验证器控制。

Initially, the Libra Blockchain only grants votes to Founding Members, entities that: (1) meet a set of predefined Founding Member eligibility criteria [46] and (2) commit a certain amount into the project. These rules help to ensure the security requirements of having a safe and live validator set. Using the Founding Member eligibility criteria ensures that the Founding Members are organizations with established reputations, making it unlikely that they would act maliciously, and suggesting that they will apply diligence in defending their validator against outside attacks.
最初， Libra Blockchain 只授予创始成员投票，这些实体:(1)符合一套预定义的创始成员资格标准[46]和(2)向项目承诺一定的金额。这些规则有助于确保拥有安全和活的验证器集的安全要求。使用创始成员资格标准确保创始成员是具有既定声誉的组织，使他们不太可能恶意行事，并建议他们将尽最大努力保护他们的验证器免受外部攻击。

While this method of assessing validator eligibility is an improvement on traditional permissioned blockchains, which usually form a set of closed business relationships, we aspire to make the Libra Blockchain fully permissionless. To do this, we plan to gradually transition to a proof-of-stake [47] system where validators are assigned voting rights proportional to the number of Libra coins they hold. This transitions the governance of the ecosystem to its users while preventing Sybil attacks [48] by requiring validators to hold a scarce resource and align their incentives with healthy system operations. This transition requires (1) the ecosystem to be sufficiently large to prevent a single bad actor from causing disruption; (2) the existence of a competitive and reliable market for delegation for users that do not wish to become validators; and (3) addressing technological and usability challenges in the staking of Libra coins.

虽然这种评估验证者资格的方法是对传统允许的区块链的改进，传统允许的区块链通常会形成一套封闭的业务关系，但我们渴望使Libra区块链完全不允许。 为了做到这一点，我们计划逐步过渡到股权证明制度[47]，验证者被分配与其持有的天秤座币数量成比例的投票权。 这将生态系统的治理过渡到其用户，同时通过要求验证者持有稀缺资源并将其激励措施与健康的系统操作相一致来防止 Sybil 攻击[48]。 这种转变要求(1)生态系统足够大，以防止单一的坏行为者造成破坏；(2)为不希望成为验证者的用户提供有竞争力和可靠的授权市场；(3)解决天秤座硬币存入过程中的技术和可用性挑战。

Another unique aspect of governance in the Libra ecosystem is that the validators form a real-world entity, the not-for-profit Libra Association, which is managed by a council of validators. A member in the association council represents each validator. A member’s voting weight in the council is the same as the validator’s voting weight in the consensus protocol. The association performs tasks such as managing the reserve, assessing the validator eligibility criteria, and guiding the open-source development of the Libra protocol.
天秤座生态系统中治理的另一个独特方面是验证者组成了一个真实的实体，即非营利Libra协会，由验证者理事会管理。协会理事会中的一名成员代表每个验证者。成员在理事会中的投票权重与共识协议中验证者的投票权重相同。协会执行管理储备、评估验证者资格标准和指导Libra协议的开源开发等任务。

Move makes it possible to encode the rules for validator management and governance as a module. Move allows the staking of coins by wrapping them in a resource that prevents access to the underlying asset. The staked resources can be used to compute the voting rights of the validators. The contract can configure the interval at which changes take effect, to reduce churn of the validator set.
Move使验证器管理和治理的规则能够编码为一个模块。Move允许通过将硬币包裹在一个阻止对底层资产访问的资源中来固定硬币。固定的资源可以用来计算验证器的投票权。合同可以配置变化生效的间隔，以减少验证器集的流失。

The Libra Association’s operation is also aided by Move. Since the association is the operator of the reserve, it can create Move modules that delegate the authority to mint and burn coins to an operational arm that interacts with authorized resellers. This operational arm can also assess if potential Founding Members meet the eligibility criteria. Move allows flexible governance mechanisms such as allowing the council to assert its authority and take back its delegated authority through a vote.
Libra协会的运营也得到了Move的帮助。由于该协会是储备的运营商，它可以创建Move模块，将铸币和烧币的权力下放给与授权经销商互动的运营部门。该运营部门还可以评估潜在的创始成员是否符合资格标准。移动允许灵活的治理机制，如允许理事会维护其权力，并通过投票收回其授权的权力。

The association has published a detailed document outlining its proposed structure [3]. All governance in the association stems from the council of validators — this council has the ultimate right to assert any authority provided to the association. Thus, the governance of the entire Libra ecosystem evolves as the validator set changes from the initial set of Founding Members to a set based on proof of stake.
该协会公布了一份详细的文件，概述了其拟议的结构[3]。协会中的所有治理都源于验证者理事会——该理事会拥有维护向协会提供的任何权威的最终权利。因此，随着验证者集从最初的创始成员集转变为基于利害关系证明的集，整个天秤座生态系统的治理也随之演变。

9.3. Validator Security and Incentives

In the initial setting, using Founding Members as validators, we believe that the institutional reputa- tion and financial incentives of each validator are sufficient to ensure that Byzantine validators control no more than f votes. In the future, however, an open system where representation is based on coin ownership will require a substantially different market design. We have started to understand the governance and equilibrium structure of a blockchain system based on stakeholdings and consumer confidence in wallets and other delegates. In the process, we have identified new market design trade-offs between the Libra approach and more established approaches, such as proof of work [49].
However, more research is needed to determine how best to maintain long-run competition in the ecosystem while ensuring the security and efficiency of the network. Furthermore, as stake-based gov- ernance introduces path dependence in influence, it is essential to explore mechanisms for protecting smaller stakeholders and service providers.
在初始环境中，使用创始成员作为验证者，我们认为每个验证者的机构声誉和财务激励足以确保拜占庭验证者控制的票数不超过 f 票。 然而，在未来，一个以硬币所有权为基础的公开系统将需要一个截然不同的市场设计。 我们已经开始理解区块链系统的治理和平衡结构，该系统基于对钱包和其他代表的信任和消费者信心。 在这个过程中，我们发现了天秤座方法和更既定的方法之间新的市场设计权衡，比如工作证明[49]。然而，需要更多的研究来确定如何在确保网络安全和效率的同时最好地保持生态系统的长期竞争。此外，由于基于股权的政府在影响力方面引入了路径依赖，因此探索保护较小的利益相关者和服务提供商的机制至关重要。

Move allows flexibility in the definition of the relevant incentive schemes such as gas pricing or staking. For example, stake slashing, a commonly discussed mechanism, could be implemented in Move by locking stake for a period of time and automatically punishing validators if they violate the rules of the LibraBFT algorithm in a way that affects safety.
Move 允许在相关激励计划的定义上有灵活性，如气体定价或固定。例如，通常讨论的削减股权机制可以通过锁定股权一段时间并自动惩罚验证者，如果他们以影响安全的方式违反了 LibraBFT 算法的规则，在 Move 中实现。

Similarly, when a validator votes in the LibraBFT algorithm, those votes can be recorded in the database. This record allows a Move module to distribute incentives based on participation in the algorithm, thereby incentivizing validators to be live. Interest generated on the Libra Reserve and gas payments can also be used as sources of incentives. Both sources are managed by Move modules, which add flexibility in their allocation. While more research is required to design an approach that will support the evolution of the Libra ecosystem, the flexibility of Move ensures that the desired approach can be implemented with few, if any, changes to the Libra protocol.
同样，当验证器在 LibraBFT 算法中投票时，这些投票可以记录在数据库中。 该记录允许Move模块基于对算法的参与来分配激励，从而激励验证器活。 天秤座储备和天然气支付产生的利息也可以作为激励来源。 这两个来源都由移动模块管理，这增加了它们分配的灵活性。 虽然需要更多的研究来设计一种支持天秤座生态系统演变的方法，但移动的灵活性确保了在对天秤座协议进行很少(如果有的话)的更改的情况下，可以实现所需的方法。

10. What's Next for Libra?

We have presented a proposal for the Libra protocol, which allows a set of validators to provide a decentralized database for tracking programmable resources. We have discussed an open-source prototype — Libra Core — of the Libra protocol and shown how the introduction of the Move programming language for smart contracts allows the protocol to implement the unique design of the Libra ecosystem.
我们已经为 Libra 协议提出了一个提案，该提案允许一组验证器提供一个用于跟踪可编程资源的去中心化数据库，我们讨论了 Libra 协议的一个开源原型- Libra Core -，并展示了智能合约的 Move 编程语言的引入如何让协议实现 Libra 生态系统的独特设计。

Both the protocol and the implementation described in this paper are currently at the prototype stage. We hope to gather feedback from the newly formed Libra Association, as well as the broader community, to turn these ideas into an open financial infrastructure. We are currently running a testnet to allow the community to experiment with this system.
本文所描述的协议和实现目前都处于原型阶段，我们希望收集来自新成立的Lib ra协会，以及更广泛的社区的反馈，将这些想法变成一个开放的金融基础设施，我们目前正在运行一个测试网，让社区对这个系统进行实验。

We are working toward an initial launch of the system, and to keep it within a manageable scope, we plan to make several simplifications in the first version. In the early days of the system, using an externally recognized set of Founding Members reduces the demands on the consensus incentive system and allows for a faster pace of updates. We anticipate using Move only for

system-defined modules, as opposed to letting users define their own modules, which allows for the Libra ecosystem to launch before the Move language is fully formed. This also allows for breaking changes to be made without compromising the flexibility that comes from defining core system behavior using Move. However, we intend for future versions of the Libra protocol to provide open access to the Move language.
我们正在努力实现系统的初步推出，为了使其保持在一个可管理的范围内，我们计划在第一个版本中进行一些简化。 在该系统的早期，使用外部认可的一组创始成员减少了对共识激励系统的需求，并允许更快的更新速度。 我们预计仅对系统定义的模块使用移动，而不是让用户定义自己的模块，这使得 Libra 生态系统能够在Move语言完全形成之前启动。 这也允许在不影响使用移动定义核心系统行为的灵活性的情况下进行打破更改。 然而，我们打算为 Libra 协议的未来版本提供对 Move 语言的开放访问。

Finally, we are currently working within the framework of the Libra Association to launch the tech- nical infrastructure behind this new ecosystem. We have published a roadmap [50] of work that we plan to contribute to support this launch. One of the top goals of the Libra Association is to migrate the Libra ecosystem to a permissionless system. We have documented the technical challenges [5] involved in making this migration. The association’ s open-source community [6] provides information about how to start using the Libra testnet, try out the Move language, and contribute to the Libra ecosystem.
最后，我们目前正在Libra协会的框架内开展工作，推出这一新生态系统背后的技术基础设施。我们已经发布了一份工作路线图[50]，我们计划为这一发布做出贡献。天秤座协会的首要目标之一是将天秤座生态系统迁移到一个无许可系统。我们已经记录了进行这种迁移所涉及的技术挑战[5]。该协会的开源社区[6]提供了关于如何开始使用天秤座测试网、尝试移动语言以及为天秤座生态系统做出贡献的信息。