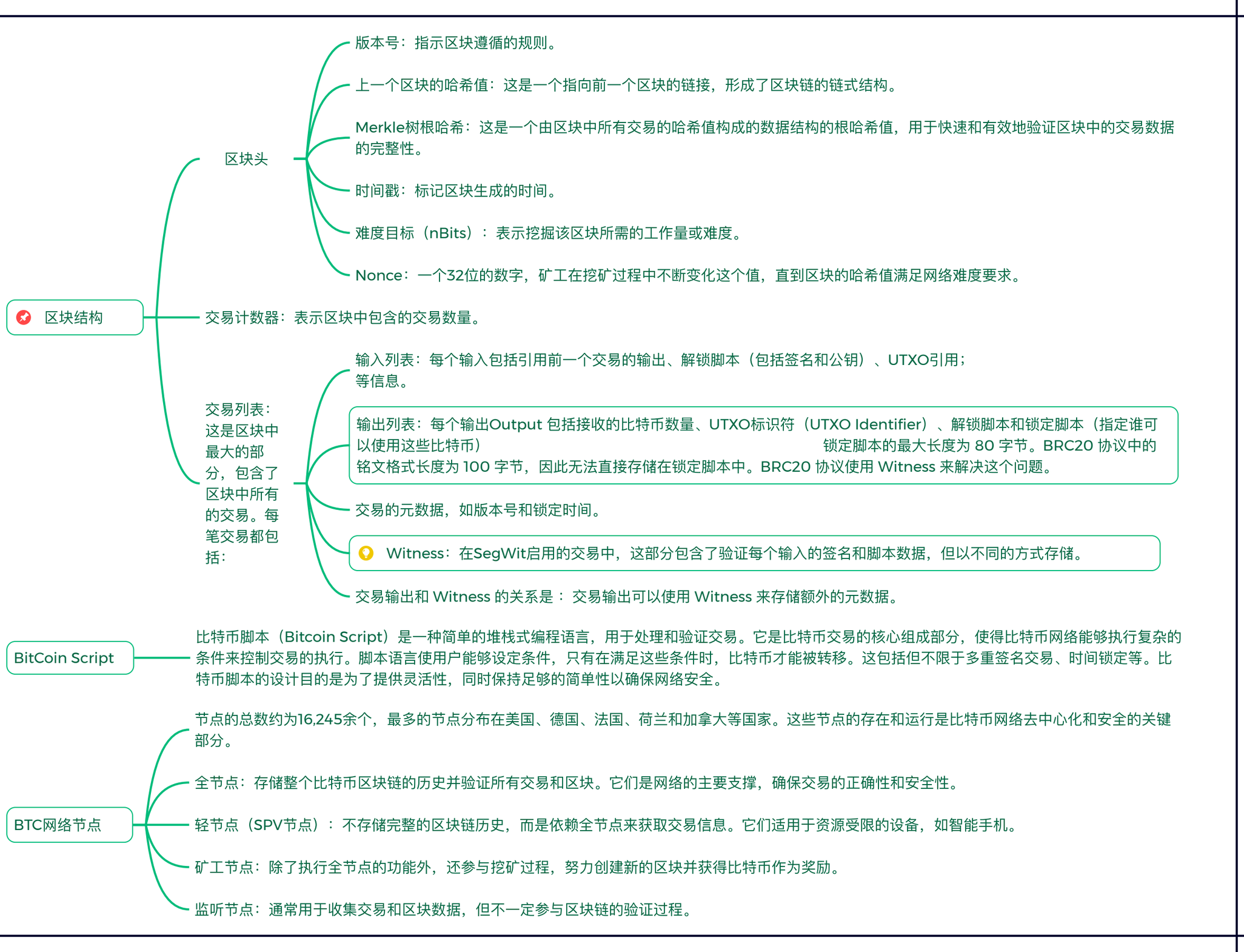
BitVM白皮书讲解

	内容	解释
贡献者	Twitter — Bitlayer @0xkevinhe — BitVM中文社区 @Bitvm_cn	BitVM中文社区致力于推广和传播BitVM技术的概念和愿景。
特别感谢	Twitter — 蓝港互动Web3研究小组 @XM_LKWeb3Team — 基础部分内容是引用蓝港互动的-比特币网络生态导航图 极客Web3 @eternal1997L — 部分内容和图片引用 - 极简解读BitVM：如何在BTC链上验证欺诈证明（执行任意计算任计算 Crypto White @biquanlibai — 部分内容引用币圈李白老师的 - BitVM 在比特币上实现智能合约 20231019	Web3是一个开源、去中心化的生态系统，吸引着众多个体自发产出丰富多样的内容。这种自主参与的特性使得Web3成为一个充满创新和协作的数字化空间。在这个生态中，人们不仅仅是消费者，更是内容的创造者，推动着生态的不断发展和演进。 BitVM技术通过允许在不改变比特币网络共识规则的前提下表达图灵完备的比特币合约，实现了执行任何可计算的函数、进行离线计算，并在链上不留下任何痕迹的特性。提供了一种全新的方式来探索复杂的区块链计算，同时保持比特币网络的原则和安全性。这使得比特币能够在无需直接修改其核心协议的情况下，扩展了更广泛的使用案例和创新可能性。 BitVM中文社区将持续关注和总结有关BitVM的最新发展，期待更多志同道合的小伙伴加入BitVM中文社区，共同建设。

基础知识 技术演进

基础知识 重要概念



作者

Robin Linus robin@zerosync.org 2023.12.12

Robin Linus — BitVM的作者，同时也是ZeroSync的创始人和核心贡献者；— Twitter

摘要

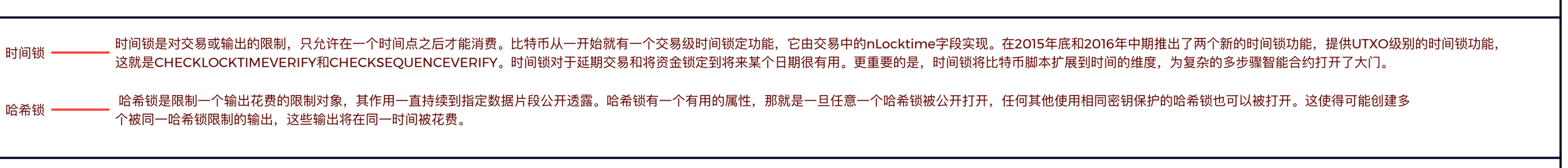
BitVM 是一种计算范式，用于表达图灵完备的比特币合约。这不需要对网络的共识规则进行任何更改。与其在比特币上执行计算，不如仅进行验证，类似于乐观Rollup。证明者提出一个声明，称给定的函数对某些特定输入计算出某个特定输出。如果该声明是假的，那么验证者可以进行简洁的欺诈证明并惩罚证明者。使用这种机制，任何可计算的函数都可以在比特币上验证。

在 Taproot 地址中提交一个大型程序需要大量的链下计算和通信，然而产生的链上足迹却很小。只要双方合作，他们可以执行任意复杂的、有状态的链下计算，而不在链上留下任何痕迹。仅在发生争议时才需要链上执行。

1 简介 Introduction

按设计，比特币的智能合约功能仅限于基本操作，如签名、时间锁和哈希锁。BitVM 为更具表现力的比特币合约和链下计算创造了新的设计空间。潜在的应用包括国际象棋、围棋或扑克等游戏，特别是在比特币合约中验证有效性证明。此外，可能还能将 BTC 桥接到外部链条、构建预测市场或模拟新的操作码。

这里提出的模型的主要缺点是它仅限于具有证明者和验证者的两方设置。另一个限制是，对于证明者和验证者而言，执行程序需要大量的链下计算和通信。然而，这些问题似乎很可能通过进一步的研究得到解决。在这项工作中，我们仅专注于两方 BitVM 的关键组成部分。



2 架构 Architecture

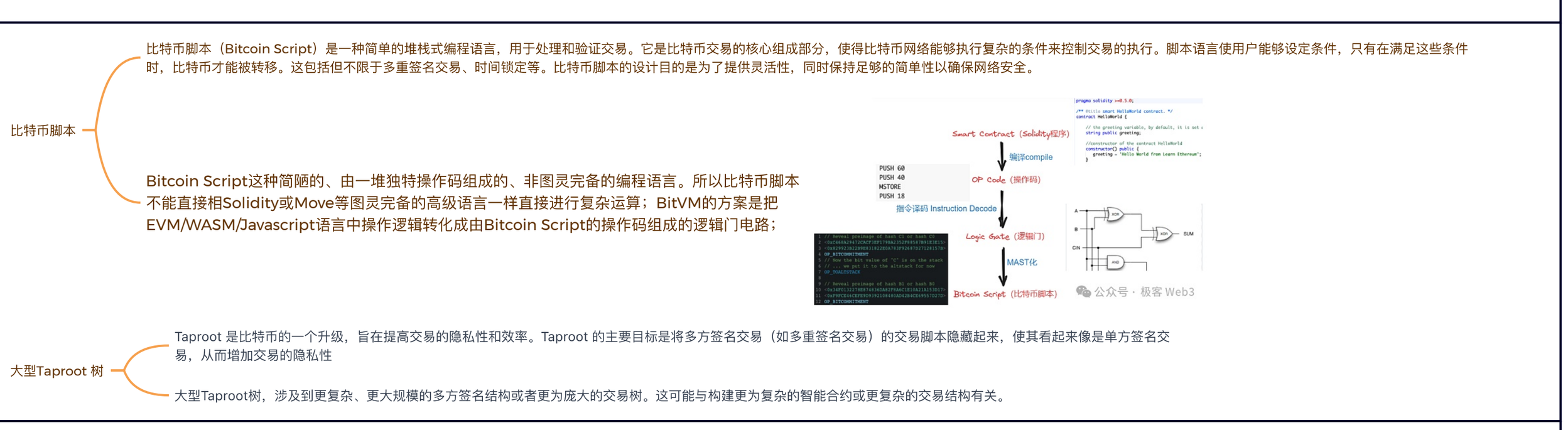
与乐观Rollup[1]和MATT提案（Merkelize All The Things）[2]类似，我们的系统基于欺诈证明和挑战-响应协议。然而，BitVM不需要对比特币的共识规则进行任何更改。底层原语相对简单。它主要基于哈希锁、时间锁和大型 Taproot 树。

证明者逐字节地承诺程序，但在链上验证所有这些将过于计算昂贵，因此验证者执行一系列精心设计的挑战，以简洁地驳斥证明者的虚假主张。证明者和验证者共同预签一系列挑战和响应交易，可用于解决以后的任何争议。

该模型旨在简单地说明，这种方法允许在比特币上进行通用计算。对于实际应用，我们应该考虑更高效的模型。

该协议很简单：首先，证明者和验证者将程序编译成一个巨大的二进制电路。证明者在一个 Taproot 地址中承诺该电路，该地址对电路中的每个逻辑门都有一个叶子脚本。此外，他们预签一系列交易，实现证明者和验证者之间的挑战-响应游戏。现在他们已经交换了所有必需的数据，所以他们可以对生成的 Taproot 地址进行链上存款。

这激活了合约，他们可以开始交换链下数据以触发电路中的状态变化。如果证明者提出任何不正确的声明，验证者可以拿走他们的存款。这保证了攻击者总是会损失他们的存款。



3 比特值承诺 Bit Value Commitment

比特值承诺是系统中最基本的组件之一。它允许证明者将特定比特的值设置为"0"或"1"。特别是，它允许证明者在不同的脚本和未使用的交易输出（UTXO）之间设置变量的值。这是关键的，因为它通过将执行运行时拆分成多个交易来扩展比特币的虚拟机（VM）。

类似于Lamport签名[3]，该承诺包含两个哈希，hash0和hash1。在稍后的某个时刻，证明者通过揭示preimage0，即hash0的原像，将比特的值设置为"0"；或者证明者通过揭示preimage1，即hash1的原像，将比特的值设置为"1"。如果在某个时刻他们同时揭示了preimage0和preimage1，那么验证者可以将它们用作欺诈证明，并拿走证明者的存款。这被称为equivocation（含糊其辞）。能够惩罚含糊其辞是使承诺具有约束力的原因 - 这是一种"基于激励的承诺"。

将比特值承诺与时间锁相结合，允许验证者在某个给定的时间范围内强制证明者决定特定比特的值。

Stack Elements

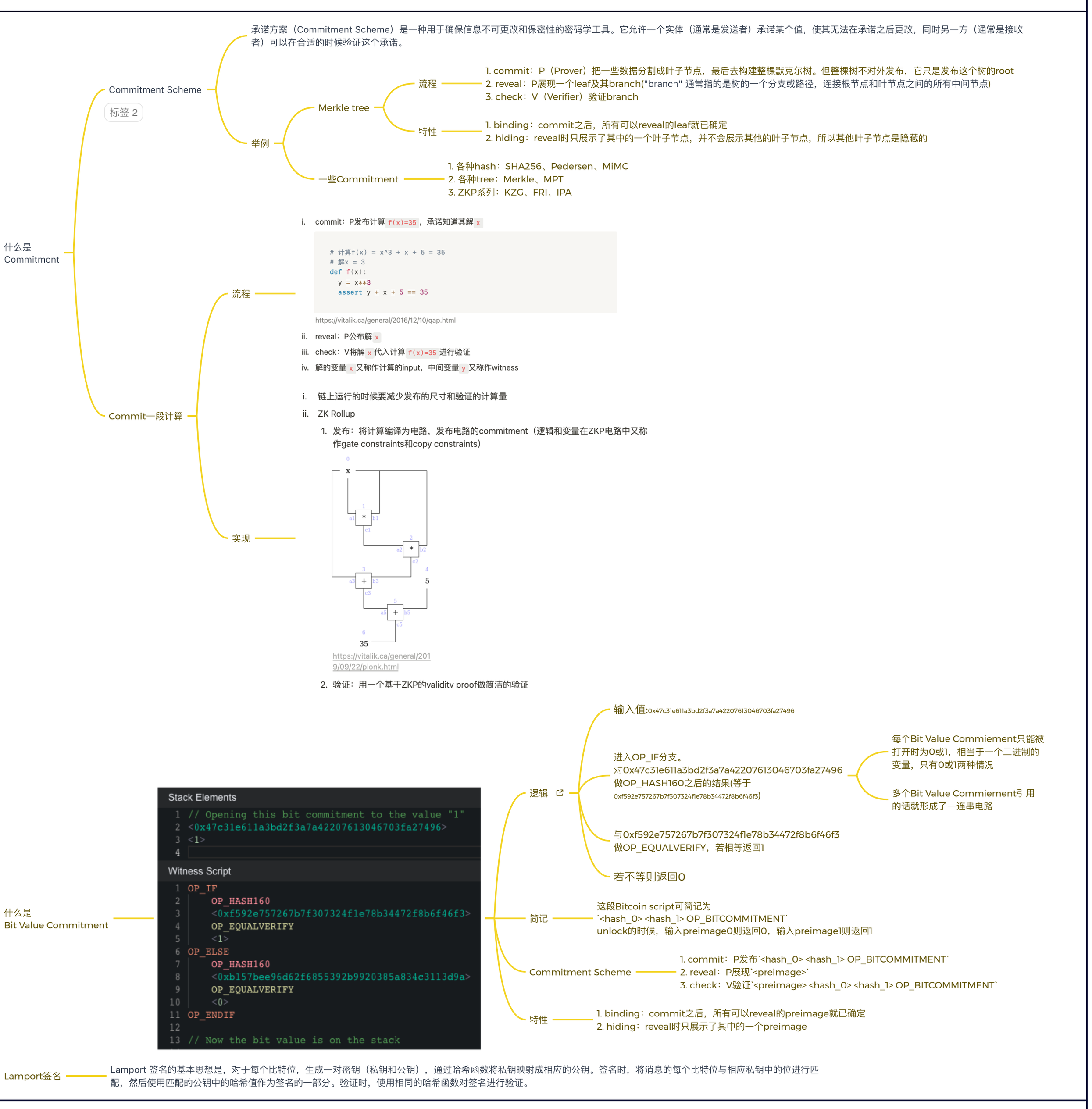
```
// Opening this bit commitment to the value "1"
<0x47c31e611a3bd2f3a7a42207613046703fa27496>
<1>

```

Witness Script

```
OP_IF
    OP_HASH160
    <0xf592e757267b7f307324f1e78b34472f8b6f46f3>
    OP_EQUALVERIFY
    <1>
OP_ELSE
    OP_HASH160
    <0xb157bee96d62f6855392b9920385a834c3113d9a>
    OP_EQUALVERIFY
    <0>
OP_ENDIF

// Now the bit value is on the stack
```

图1：一个用于1比特承诺的具体实现。为了解锁此脚本，证明者必须揭示hash0或hash1的原像中的一个。在此示例执行中，证明者揭示了hash1，并将比特的值设置为"1"。我们可以拥有这个承诺的副本，以强制在不同脚本之间执行特定值。

为简单起见，从这里开始，我们假设存在一个名为OP_BITCOMMITMENT的操作码，它是上述脚本的缩写。该操作码消耗两个哈希和其中一个哈希的原像。根据原像匹配的哈希，它将比特值放在堆栈上。



4 逻辑门承诺 Logic Gate Commitment

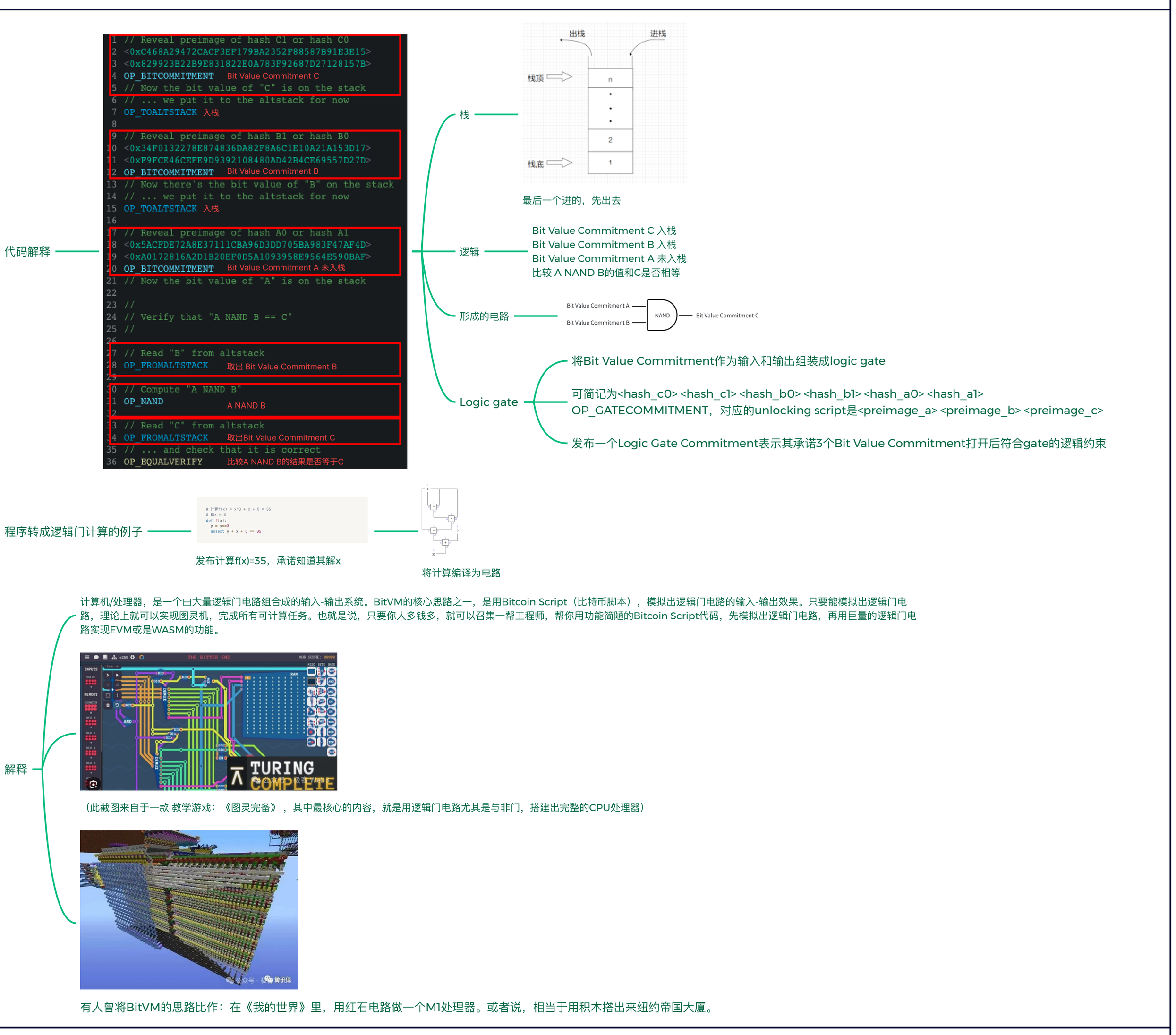
任何可计算函数都可以表示为布尔电路。与非门（NAND）是一种通用逻辑门，所以任何布尔函数都可以由它组成。为保持模型简单，展示了我们的方法适用于简单的与非门。此外，我们展示了如何任意组合门。这一起证明了 BitVM 可以表达任何电路。

与非门承诺的实现很简单。它包含代表两个输入的两个位承诺和代表输出的第三个位承诺。脚本计算两个输入的与非值，以确保它与承诺的输出位相匹配。

```
// Reveal preimage of hash C1 or hash C0
<0xC468A29472CACF3EF179BA2352F88587B91E3E15>
<0x829923B22B9E831822E0A783F92687D27128157B>
OP_BITCOMMITMENT
// Now the bit value of "C" is on the stack
// ... we put it to the altstack for now
OP_TOALTSTACK

// Reveal preimage of hash B1 or hash B0
<0x34F0132278E874836DA82F8A6C1E10A21A153D17>
<0xF9FCE46CEFE9D9392108480AD42B4CE69557D27D>
OP_BITCOMMITMENT
// Now there's the bit value of "B" on the stack
// ... we put it to the altstack for now
OP_TOALTSTACK

// Reveal preimage of hash A0 or hash A1
<0x5ACFDE72A8E37111CBA96D3DD705BA983F47AF4D>
<0xA0172816A2D1B20EF0D5A1093958E9564E590BAF>
OP_BITCOMMITMENT
// Now the bit value of "A" is on the stack

//
// Verify that "A NAND B == C"
//

// Read "B" from altstack
OP_FROMALTSTACK

// Compute "A NAND B"
OP_NAND

// Read "C" from altstack
OP_FROMALTSTACK
// ... and check that it is correct
OP_EQUALVERIFY
```

图2：NAND操作的逻辑门承诺。执行此脚本需要揭示比特承诺A、B和C的值，使得A NAND B = C成立。

（在这里，我们为简单起见假设存在用于OP_NAND的操作码。实际上它并不存在，然而，可以使用OP_BOOLAND和OP_NOT轻松实现。）



5 二进制电路承诺 Binary Circuit Commitment

上一节，定义了与非门承诺。通过组合门承诺来表达任何电路。执行的每一步都在一个 Tapleaf 中提交。它们都合并到同一个 Taproot 地址中，使得证明者可以执行电路中的任何门。执行一个门需要证明者打开相应的门承诺，并为其输入和输出位设置值。 Taptree 可能变得非常庞大，拥有十亿个 Tapleaf 脚本，但它在链上的足迹却很小。

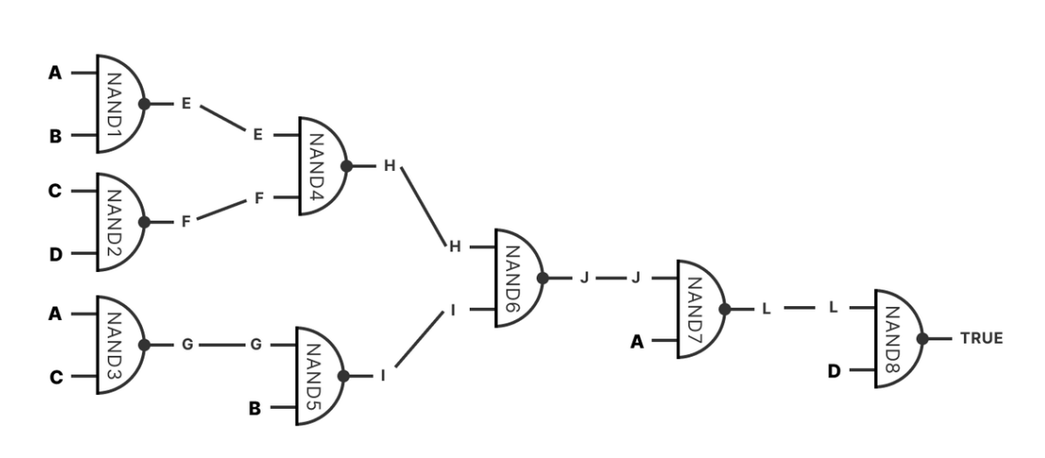


图3：一个随机示例电路，有8个不同的与非门和4个输入A、B、C和D。使用数十亿个门将允许我们定义基本上任何函数。

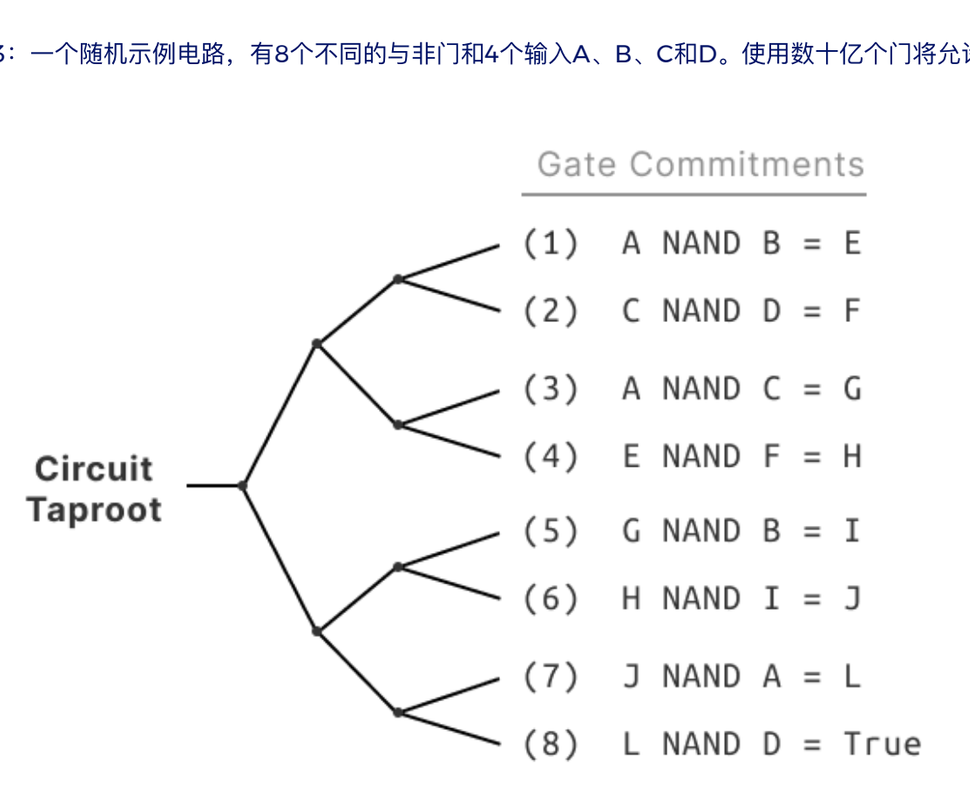
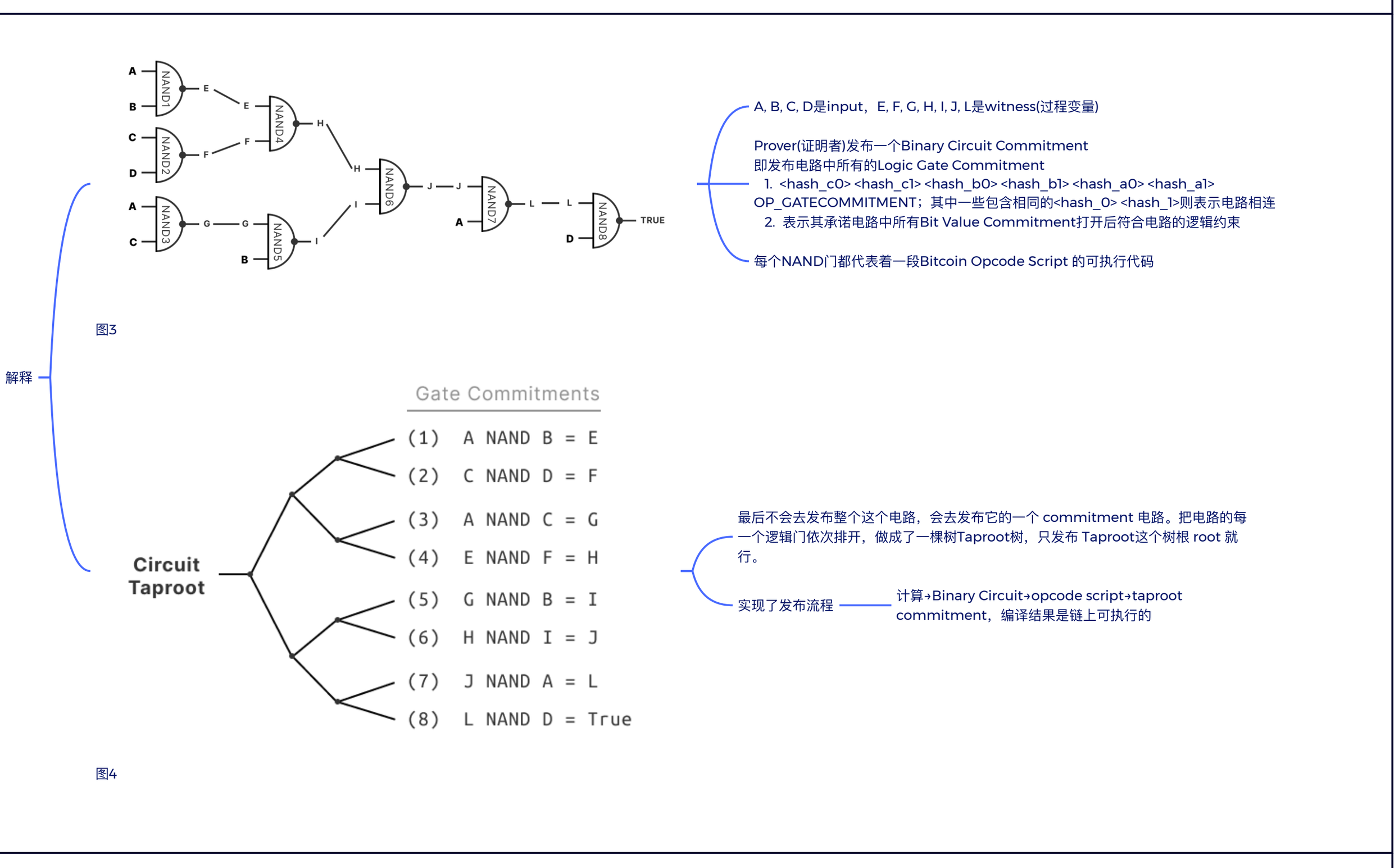


图4：对于每个门，证明者的 Taproot 地址包含一个带有相应门承诺的叶脚本。这使证明者可以在以后的任何时候设置电路的输入值（这里是A、B、C和D）。



6 挑战与响应 Challenges and Responses

承诺一个电路是不够的。为了驳斥一个不正确的主张，验证者必须能够挑战证明者的声明。这是通过他们在设置期间预先签署一系列交易来实现的。这些交易像挑战 → 响应 → 挑战 → 响应......这样链接起来。如果双方中的一方停止参与，那么在一定的超时后，另一方将赢得挑战并可以拿走双方的存款。只要双方合作，他们可以用2-of-2签名共同解决任何合约。以下机制仅在欺诈情况下需要。

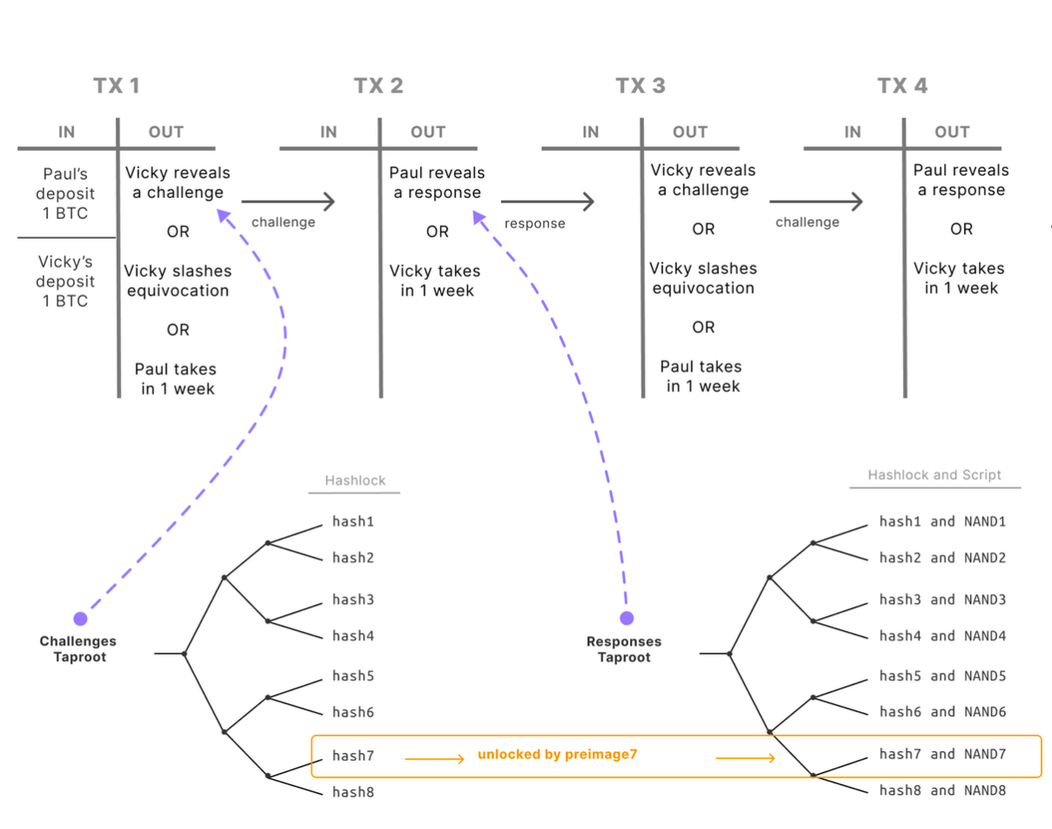


图5：在设置期间生成的一系列预签交易，用于执行多轮挑战和响应。

Vicky通过打开她的Tapscript叶中的一个哈希锁来选择一个挑战。这为Paul解锁了一个特定的 Tapscript，并迫使他执行它。脚本迫使 Paul 揭示 Vicky 所挑战的门承诺。通过重复这个程序几轮查询，任何不一致的主张都可以迅速被驳斥。

如果证明者停止与验证者的链下合作，验证者需要一种方法在链上强迫他的行动。验证者通过解锁一个哈希锁来做到这一点：证明者的UTXO中的每个与非 Tapleaf 只有在证明者知道验证者持有的一个原像时才能被消费。因此，证明者可以通过揭示其输入和输出来证明给定的 Tapleaf 正确执行，但仅当验证者通过揭示守卫该 Tapleaf 的哈希的原像为他"解锁"时才能这样做。应用二分查找，验证者在log(N)轮挑战和响应后就可以快速识别证明者的错误。

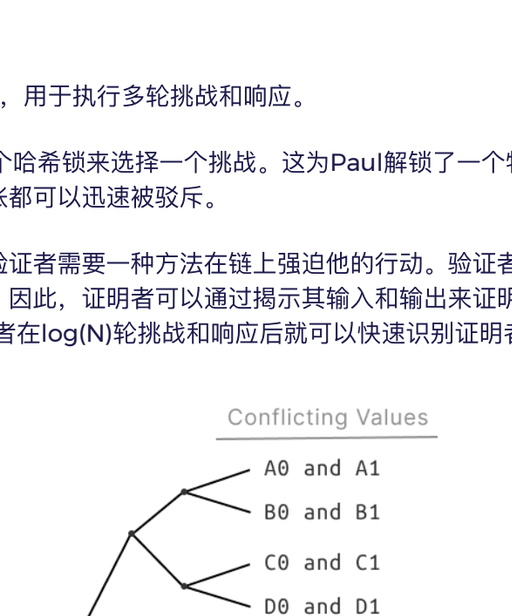
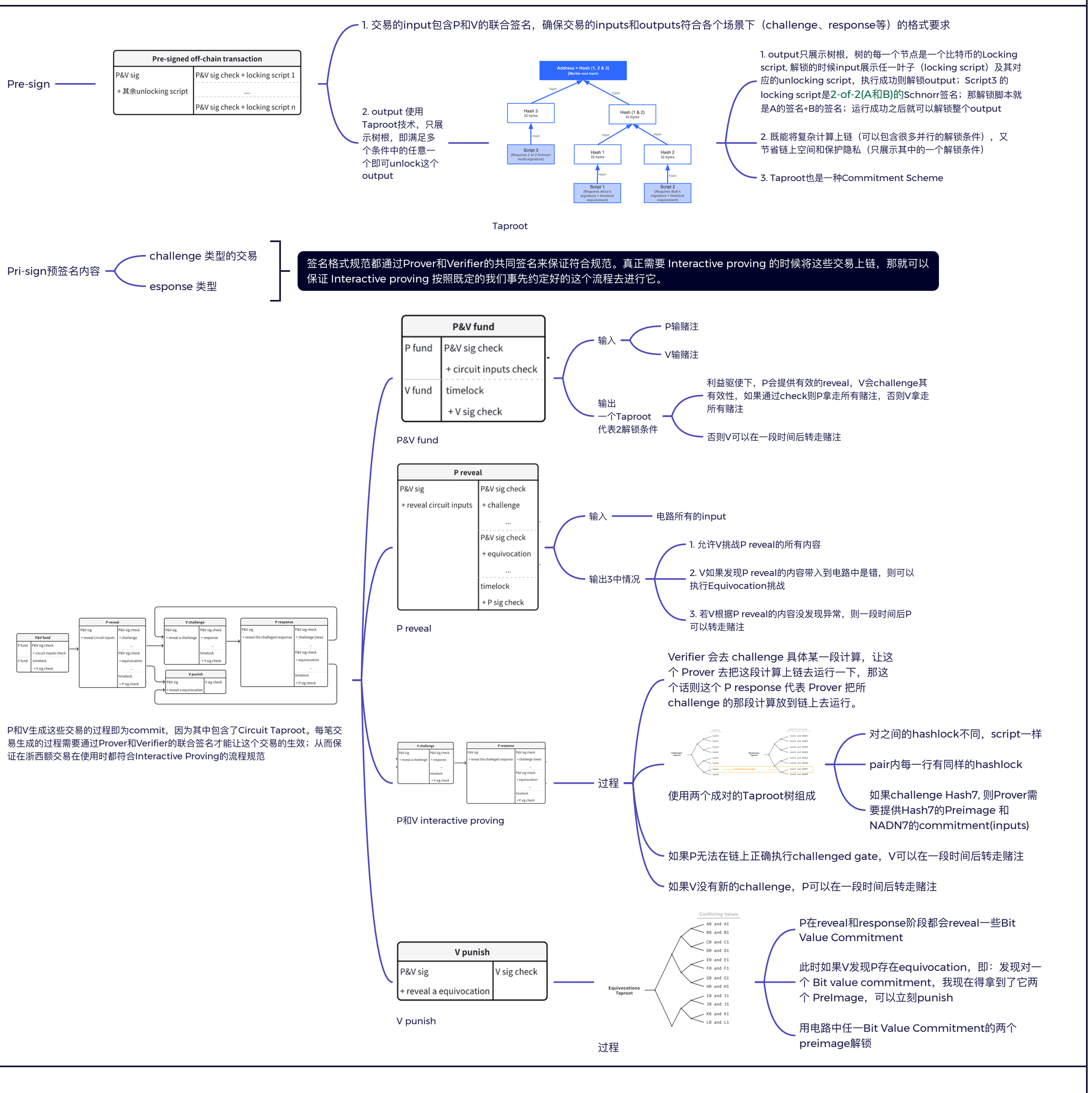


图6：在每次响应后，Vicky可以惩罚含糊其辞。如果 Paul 曾经为变量揭示两个矛盾的值，那么 Vicky 立即赢得挑战，并被允许拿走他的存款。Vicky 通过揭示 Paul 的任何一个位承诺的两个原像来证明 Paul 的含糊其辞。



7 输入与输出 Inputs and Outputs

证明者可以通过揭示相应的比特承诺来设置输入。理想情况下，他们会在链外揭示这些承诺，以最小化在链上的占用。在非合作情况下，验证者可以强制证明者在链上揭示他们的输入。

通过预先交换数据但加密的方式，可以处理大量数据。这样证明者可以在之后的某个时刻揭示解密密钥。

多方输入也是可能的。门可以有来自双方的位承诺。

8 局限性与展望 Limitations and Outlook

在简单的NAND电路中表达函数是低效的。通过使用更高级的操作码，程序可以更有效地表达。例如，比特币脚本支持添加32位数字，因此我们无需为此使用二进制电路。我们还可以拥有更大的比特承诺，例如，可以在单个哈希中承诺32位。此外，脚本的大小可以达到约4 MB。因此，我们可以在每个叶子脚本中实现远远超过一个NAND指令。

这里提出的模型仅限于两方。然而，可能可以建立双向通道，并将它们链式连接以形成类似于闪电网络的网络。探索双方设置可能会产生一些有趣的泛化可能性。例如，我们可以探索网络的1对n星型拓扑结构。另一个研究问题是我们是否可以将我们的模型应用于n-of-n设置，并创建更复杂的通道工厂。此外，我们还可以将我们的系统与不同的链下协议结合使用，例如闪电网络或rollups。

其他研究方向包括跨应用内存、如何对刻在链上的任意数据进行陈述，或链下可编程电路，即链下虚拟机。还有可能应用更复杂的采样技术，类似于STARKs，以在单一轮中检查电路。

下一个重要的里程碑是完成具体的BitVM设计和实现，以及Tree++，一个用于编写和调试比特币合约的高级语言。

9 结论 Conclusion

比特币在编码欺诈证明的大型 Taptrees 中可以验证任何程序的执行，因此在某种意义上是图灵完备的。这里概述的模型的一个主要限制是它仅限于两方设置。希望这可以在后续工作中得到泛化。

致谢

特别感谢Super Testnet和Sam Parker，他始终拒绝相信比特币不会是图灵完备的。

参考文献 References

[1] Ethereum Research. Optimistic rollups. https://ethereum.org/en/developers/docs/scaling/optimistic-rollups/, 2022.
[2] Salvatore Ingala. Merkleize all the things. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2022-November/021182.html, 2022.
[3] Jeremy Rubin. CheckSigFromStack for 5 Byte Values. https://rubin.io/blog/2021/07/02/signing-5-bytes, 2021.