西安交通大學

硕士研究生中期进展报告

学　　号：	3122358153
姓　　名：	王璟
导　　师：	齐赛宇
论文题目：	基于模糊测试路径探索的区块链共识算法研究与实现
学科专业：	软件工程
学　　院：	软件学院
填写时间：	2024　年　7 月　1 日

西安交通大学研究生院制

硕士研究生中期进展报告填写说明及管理规定

一、硕士学位论文中期考核要求在校硕士生必须在入学第四学期末（两年毕业试点学院的硕士生应在第三学期末）完成。

二、硕士研究生在完成了一定论文工作的基础上，填写完成《硕士研究生中期进展报告》。

三、《硕士研究生中期进展报告》完成以后，应组织公开的中期考核报告会。

四、中期考核由学院负责统一组织，考核专家组一般由 5 名副高以上（含副高）人员组成。考核专家主要依据硕士研究生的论文课题进展情况进行考核，同时可参阅其课程学习和选题报告情况。中期考核结束后，考核专家应在本表中填写考核评价结果、评语和论文修改意见。

五、《硕士研究生中期进展报告》必须采用 A4 纸双面打印，左侧装订成册，各栏空格不够时，请自行加页。本表可在研究生院主页 http://gs.xjtu.edu.cn/下载。

六、《硕士研究生中期进展报告》由学院归档。

论文题目	基于模糊测试路径探索的区块链共识算法研究与实现

一、研究内容简介（300～500 字）

区块链（Blockchain）是一种底层技术和基础架构，本质上是一个存放在非安全环境中的分布式的、去中心化的、篡改难度极高的数据库系统，它以块的形式存储数据，采用密码学方法保证数据不被篡改，采用共识算法对新增数据达成共识。区块链技术建立了新的信任机制，允许在没有权威节点的去中心化情况下，各网络节点之间达成可信共识，是一项从思想到技术的重大飞跃。

共识算法是区块链技术最核心，也是整个技术发展和学术界最热衷的领域。目前常见的共识算法有工作量证明（PoW，Proof-of-Work）、权益证明（PoS，Proof-of-Stake）、委托权益证明（DPoS，Delegated Proof-of-Stake）等。然而，传统 PoW 需要较高的处理器能力和能耗，环境成本高，且这些算力只用于挖矿，并没有其他用途。因此，许多研究致力于将区块链与其他技术相结合，以利用被浪费的算力，同时还能激励其他技术的发展。

与此同时，模糊测试（Fuzzing）作为一种软件测试技术，其核心思想是将自动或半自动生成的随机数据输入到待测程序中，监视异常以发现可能存在的程序漏洞，包括但不限于处理异常输入时的缓冲区溢出、内存泄漏和崩溃。现有的 Fuzzing 方法在面对复杂系统时，容易出现饱和效应（Saturation effects），指持续使用同一刺激材料导致刺激效果递减的心理现象，这使得在模糊测试中，测试人员难以持续高效地发现新漏洞。

因此，本课题提出了一种新的工作量证明，称为 Proof-of-Fuzzing，将传统 PoW 和 Fuzzing 结合，利用被浪费的算力，并提供一个漏洞奖励平台，为有检测代码漏洞需求的人提供服务，同时也通过奖励机制激励矿工发现更多漏洞。

二、研究工作进展（已完成的主要工作及已取得的成绩。3000～5000 字）

1、将模糊测试过程转化为 Proof-of-Fuzzing 共识算法

1.1 Proof-of-Fuzzing 共识算法的设计

考虑到要将一个新的共识算法引入区块链网络，必须遵守以下几个条件：

（1） 这个问题必须是复杂的，需要一定的计算量，才能保证矿工进行了一些实际的工作，从而能够获得与区块挖掘相关的奖励；

（2） 为了保证区块链的完整性，必须将前一个区块的 hash 值作为问题的变量引入；

（3） 挖矿方案必须有一个竞争性的组成部分，这样第一个解决问题的矿工（或者提供最佳解决方案的矿工）就是挖出区块并获得奖励的矿工；

（4） 给定一个问题解决方案，必须很容易验证该解决方案的有效性并评估其质量；

（5） 一旦某个矿工成功将一个新区块添加到当前区块链中，其他矿工正在挖掘的所有其他潜在区块都必须被丢弃。这保证了矿工不能“保存区块”用作之后的挖矿结果。

漏洞奖励计划（Vulnerability Reward Program，VRP）是一种激励安全研究人员发现和报告软件漏洞的机制，通过这种机制，软件开发者可以及时修复漏洞，提高软件的安全性和稳定性，同时漏洞发现者也可以获得奖励。漏洞奖励计划是目前很多科技公司查找自家漏洞的主要方法之一，包括 Google、Apple、Microsoft 等公司。在 2023 年内，Google 向来自 68 个国家的 632 名研究人员支付了 1000 万美元（约合人民币 7196 万元），以奖励他们发现并报告 Google 旗下产品和服务的安全漏洞。

本课题受漏洞奖励计划的启发，我们建议矿工解决的难题是在一定时间内，对程序提供者发布的被测程序进行持续性的模糊测试以达到覆盖率要求，并要求计算出的 hash 值在给定区间内，Fuzzing 本身就是一种计算昂贵的任务，因此满足条件 1。矿工在计算本次执行的 hash 值时，需要包含前一个区块的 hash 值，因此满足条件 2。本共识算法要求计算出的 hash 值在一个动态调整的区间中，第一个满足要求的矿工获得下一个区块的记账权，因此满足条件 3。每次执行窗口结束后，计算出符合要求的 hash 值的矿工广播生成的新区块，由其他矿工验证它们的合法性，新区块的验证与传统 PoW 的验证过程类似，满足条件 4。一旦确定获胜的矿工，该区块将被添加到区块链末端，那么最后一个区块的 hash 值就发生了变化，成为待开采新区块的前一个区块，计算新 hash 值中的 pre_hash 发生变化，矿工们将会在这个新的 pre_hash 基础上继续挖矿，满足条件 5。

1.2 执行路径的获取

本课题选用 AFL（American Fuzzing Loop）作为模糊测试工具，AFL 是一种主流的基于覆盖引导（Coverage-guided）的灰盒模糊器（Fuzzer），它通过记录输入用例的代码覆盖率，从而调整输入用例以提高覆盖率，增加发现漏洞的概率。AFL 维护一个测试用例队列，每次选择一个测试用例进行多轮变异（Mutation），如位翻转、替换等，变异后的测试用例将作为被测程序的输入。之后 Fuzzer 便会向被测程序发送执行信号，等待执行完成后收集代码覆盖率和退出状态等信息。在本课题的设计中，每位矿工在给定时间内执行被测程序后都需要给出执行用例和它对应的执行路径，在 AFL 中并没有直接输出路径的功能，因此为实现这一目标，需要对 AFL 的源码进行修改。

AFL 使用编译时插桩来跟踪被测程序的执行过程，它通过记录被测程序中基本块（Basic Block）之间的跳转来实现这一目的。AFL 插桩代码使用的语言是 AT&T 风格的汇编语言，桩被插入到已编译程序中的特定位置，在以下几个地方插桩：函数入口点、条件跳转指令后、指令的标签后。桩指令完成的任务可以概括为图 1。

```
cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
```

图 1 AFL 插桩核心代码逻辑

AFL 在插桩时为每一个基本块赋值随机数（即图 1 中的 COMPILE_TIME_RANDOM），作为它的标识 ID，两个 ID 对即表示控制流转换（我们称之为边）。当被测程序运行到这个基本块时，该块即被命中（hit），AFL 将当前块和前一块异或后保存到共享内存（Fuzzer 和被测程序间共享，即图 1 中的 shared_mem）中，最后将当前块右移一位（为了区分两个块之间不同方向的路径），完成对两个块之间的边的标记，因此获取执行路径就转换为获取这些被命中的块的 ID。根据 AFL 源码中桩代码的逻辑，找到对图 1 中的 cur_location 和 pre_location 进行异或处理的部分，在这部分中添加汇编代码，把基本块的 ID 输出到指定文件中。

用一个简单的 C 程序测试得到的路径信息文件的一部分如图 2 所示，每行表示一个测试用例的输入和路径，输入用二进制表示，是 AFL 经过不断变异后的结果，路径为被命中的基本块的 ID 的顺序输出，可以看到不同的输入下的执行路径也不完全一样。在实际的运行环境中，为保证公平性，矿工用来 Fuzzing 的种子文件是随机分配的。

```
用例：b'k\x89\x00\x01'；路径：29084->20050->4374
用例：b'kljhhoaj'；路径：29084->20050->58506->15170->4374
用例：b'ql\x84'；路径：29084->20050->58506->4374
用例：b'\x00\x01\x00\x00\x00'；路径：29084->20050->4374
用例：b'{{'；路径：29084->20050->4374
用例：b'q^Qooooooooooooooo '；路径：29084->20050->58506->15170->4374
用例：b'kkl\x80\xffj'；路径：29084->20050->58506->15170->4374
用例：b'cj'；路径：29084->20050->4374
用例：b'\xe4\x7faj'；路径：29084->20050->58506->4374
用例：b'ulqhooaj'；路径：29084->20050->58506->15170->4374
用例：b'.\x7f'；路径：29084->20050->4374
用例：b'\xe9\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\x00\x01'；路径：29084->20050->58506->15170->4374
用例：b'\x01'；路径：29084->20050->4374
用例：b':\x1d\n\xf7\x7f\x04\x00c\x1a'；路径：29084->20050->4374
用例：b'@\x00 \x00roaj'；路径：29084->20050->4374
用例：b'\x00\x00\x00\x00\x00\x00\x00\x00\x02'；路径：29084->20050->4374
用例：b'\n\xf3\x00\x99z\x17h'；路径：29084->20050->4374
用例：b'\x80\x00Hh`^'；路径：29084->20050->4374
用例：b'\x04'；路径：29084->20050->4374
```

图 2 执行路径输出文件

1.3 执行窗口的实现

由于每个人每次执行代码的时间都是不一样的，为了公平判定矿工对该被测程序漏洞挖掘的贡献度以决定记账权的分配，本课题设计了一个执行窗口，该窗口以一定的时间（目前设置为 1s）为界，统计每个矿工在窗口中执行的用例输入、路径信息等等。

设置固定的执行时间会出现窗口结束时本次测试用例的执行路径并未跑完的情况，除此之外，窗口结束后需要计算 hash 值，之后每位矿工都需要对其他矿工广播的新区块进行验证处理，因此可在每轮窗口结束时挂起 Fuzzing 进程，当处理完后再从挂起的地方恢复执行。在实现该功能时，采用 setitimer 定时器的方法来控制执行窗口的时间，倒计时结束后触发 SIGALRM 信号，Fuzzer 收到该信号后向正在进行模糊测试的子进程发起 SIGSTOP 信号，使子进程挂起，此时即可对这次执行窗口中的用例和路径等信息做统计，计算 Hash 值并签名，最后将新区块进行广播。处理完这些后，向子进程发起 SIGCONT 信号，恢复执行。

hash 计算的过程如图 3 所示，采用 sha256 加密算法来进行 hash 计算。其中 Hash N 表示区块链中前一个区块的 hash 值，矿工会从待确认的交易池中选择本次要打包的交易，Nonce 是由随机数生成器生成的随机数，矿工在该次执行窗口中执行的每条测试用例（case）都记录为（path，payload，iscrash）这样的三元组，其中 path 为执行路径，payload 为用例的输入，iscrash 表示该用例是否被 AFL 标记为漏洞，Timestamp 为矿工打包这个区块的时间戳。值得注意的是，只有计算出符合要求的 Hash 值的矿工才需要广播自己的结果。

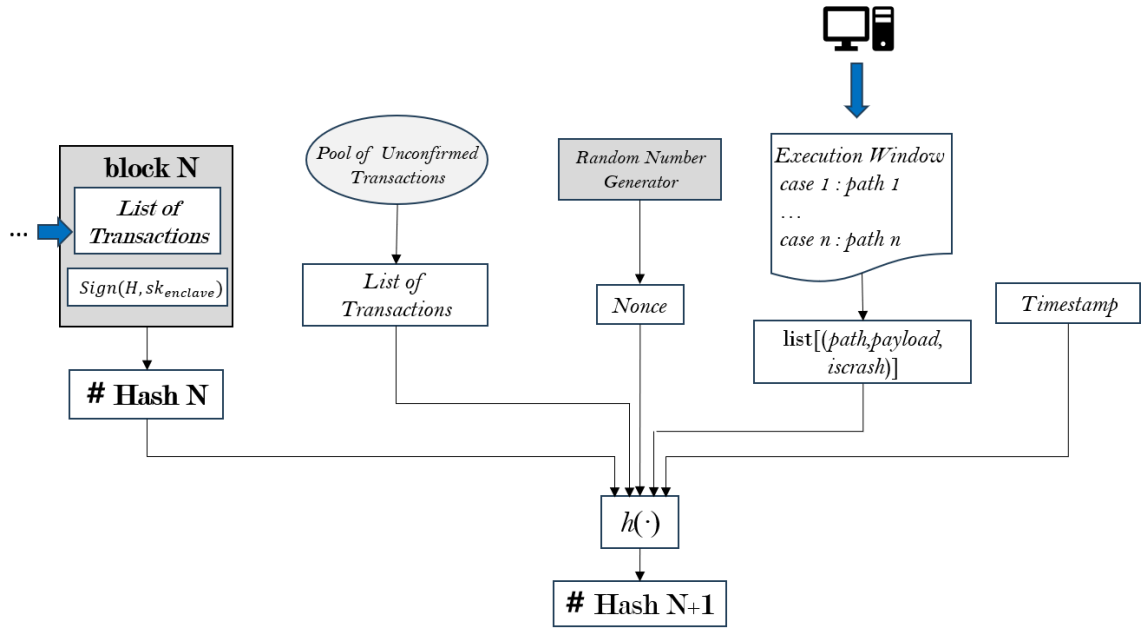

图 3 Hash 计算过程

考虑到执行窗口的部分可能会受到恶意矿工的攻击，使得该矿工可以不遵守执行窗口的时间限制，从而提交更多的执行路径，这会破坏挖矿的公平性，为了确保矿工可信执行了挂起操作，选择将执行逻辑放入到 Intel SGX（Intel Software guard extensions）中以确保定时挂起，SGX 是 Intel 架构新的扩展，在原有架构的基础上增加了一组新的指令集和内存访问机制，这些扩展允许应用程序实现一个被称为 enclave 的容器，在应用程序的地址空间中划分出一块被保护的区域，为容器内的代码和数据提供机密性和完整性的保护，免受恶意软件的破坏。矿工在执行窗口结束后计算出的 hash 值和该矿工的 enclave 信息拼接后用私钥签名，得到结果为$Sign(H, sk_{enclave})$，将打包好的新区块广播到全网。

1.4 记账权和漏洞验证

矿工们接收到其他人广播的区块后，使用对应公钥验证接收到的第一个区块的 enclave 的身份信息的正确性，通过验证后再校验 hash 的正确性等，如果全部通过校验就将它添加到本地区块链末端，更新当前链为最长链，并将该新区块发送给其他节点，如果该区块没有通过验证则拒绝它。由于路径 hash 值的分布是不均匀的，因此本课题引入了动态可变映射的 hash 区间，通过反馈的方式动态调整该 hash 区间，也就是动态调整挖矿难度，确保在固定时间块内有人可以获得记账权，即所探索的路径的 hash 落到该区间内。

如果该矿工提交的用例中存在 iscrash 为 true 的情况，那么验证的矿工需要将该用例执行一次，与提交的路径进行对比，验证通过则认为这是一个潜在的漏洞，后续会提交给开发人员来判定这是否是有效漏洞。

1.5 奖励机制的设计

本课题的奖励机制包括三部分：基本奖励、漏洞奖励和新路径奖励。基本奖励即成功挖矿的奖励，与传统 PoW 类似，该奖励只发放给获得记账权的矿工，且随模糊测试难度不断调整。基本奖励计算方法如下所示：

$$Reward = k * D$$

其中，D为模糊测试难度，它的计算方法如下所示：

$$D = w_1 \cdot C + w_2 \cdot R + w_3 \cdot P$$

其中，C为代码覆盖率因子，R为资源消耗因子，P为程序复杂度因子，w_1, w_2, w_3分别为各因子对应的权重（初始值为 0.4，0.3，0.3，之后会进行调整）。

代码覆盖率因子计算方法如下所示：

$$C = \frac{LC + BC + FC}{3}$$

其中，LC为行覆盖率，BC为分支覆盖率，FC为函数覆盖率。

资源消耗因子计算方法如下所示：

$$R = \frac{T}{T_{max}} + \frac{M}{M_{max}}$$

其中，T为模糊测试执行时间，T_{max}为模糊测试允许执行的最大时间，M为模糊测试占用的内存，M_{max}为允许的最大内存。

程序复杂度因子计算方法如下所示：

$$P = \frac{L}{L_{max}} + \frac{F}{F_{max}} + \frac{D_{num}}{D_{max}}$$

其中，L为代码行数，L_{max}为总代码行数，F为函数数量，F_{max}为总函数数量，D_{num}为依赖库数量，D_{max}为总依赖库数量。

代码覆盖率包括行覆盖率、分支覆盖率、函数覆盖率，资源消耗即模糊测试过程中的执行时间、内存占用等，程序复杂度通过目标程序的代码行数、函数数量、依赖关系等来判定。代码覆盖率越高、执行时间越长、内存占用越大、程序越复杂，模糊测试难度就越大，奖励也就会相应提高。基本奖励由系统发放。

漏洞奖励是给发现并提交有效漏洞的矿工的额外奖励，该奖励是由各程序提供方发放。奖励金额可以根据漏洞的严重程度进行评估。漏洞的严重程度S_{bug}通过通用漏洞评分系统（Common Vulnerability Scoring System，CVSS）来判定，CVSS 是一个开放的行业标准，用于评估计算机系统安全漏洞的严重性，它的分值范围是 0.0~10.0，数字越大代表漏洞的严重程度越高，漏洞奖励与S_{bug}成正比。新路径奖励根据该矿工发现的新路径占本次窗口发现的所有新路径的比重来分发。最终奖励为基本奖励、漏洞奖励和新路径奖励之和。

2、系统设计和实现

2.1 系统架构

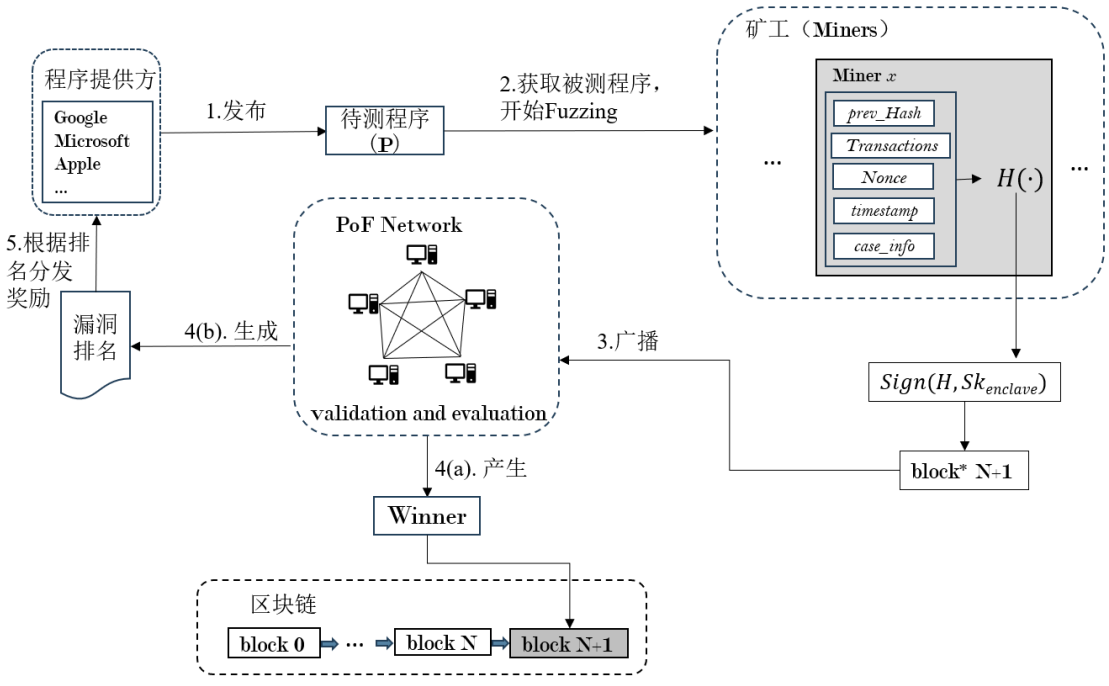

图 4 系统架构图

本课题将实现一个基于 Proof-of-Fuzzing 的分布式漏洞挖掘系统，系统架构如图 4 所示。系统的使用者包括程序提供方和漏洞挖掘者（也就是矿工）。程序提供方即漏洞奖励计划中提供待测程序的公司（如 Google、Microsoft、Apple 等），这些提供方发布待测程序后，矿工们即可获取待测程序开始 Fuzzing，经过一个执行窗口后，Fuzzing 被挂起，计算出符合要求的 hash 值的矿工广播新区块，其他矿工在接收后即进行验证和评估，对记账权达成共识，产生本轮执行窗口的 winner 和漏洞排名，系统给 winner 发放基本奖励，程序提供方根据生成的排名分发漏洞奖励。

2.2 功能模块设计

整个系统分为 6 个模块：待测程序发布模块、模糊测试模块、消息发送接收模块、工作量和漏洞验证模块、区块生成和同步模块以及奖励发放模块。每个模块具体包含的功能如下所示：

（1） 待测程序发布模块：由程序提供方发布需要漏洞挖掘的程序源码、覆盖率目标等；

（2） 模糊测试模块：矿工获取任务，在执行窗口中进行模糊测试，生成运行的测试用例、执行路径、是否发现漏洞等信息，计算 hash 后对它签名；

（3） 消息发送和接收模块：矿工在执行窗口结束后广播本轮创建的新区块，其他矿工校验和评估后将结果继续广播；交易者将发起的交易广播等等；

（4） 工作量和漏洞验证模块：矿工校验其他人发送的新区块中的签名是否正确，验证 hash 是否合法等，对记账权的归属达成共识；验证漏洞是否真实有效；

（5） 区块生成和同步模块：矿工将校验通过的新区块添加到本地区块链末端，更新当前链为最长链，并将该新区块发送给其他节点；动态调整下一轮的 hash 区间；

（6） 奖励发放模块：给挖矿成功的矿工发放奖励，根据计算出的模糊测试难度、发现的漏洞的严重程度、路径贡献度给成功挖掘漏洞的矿工发放奖励。

三、下一步的工作计划

1、完成执行窗口的实现
2、完善动态调整挖矿难度算法、完成代码实现
3、完善奖励机制设计、完成代码实现
4、搭建区块链系统，将所有功能都整合起来
5、对系统进行评估测试
6、撰写论文，尝试投稿

四、指导教师评语

该生在前期的工作中能充分分析课题任务需求，掌握区块链基础知识，系统设计合理，为下一步工作的展开做了充分准备。期间该生阅读了大量的文献资料，学习态度认真，能按要求积极推进工作，研究工作有一定的实际意义，课题进展符合预期计划。

签名：  日期：2024 年 7 月 8 日

五、中期考核结果（请在相应等级后的“（）”内打“ √ ”，并给出评语和修改意见）

考核结果:　　优秀（　）；　　合格（　）；　不合格（　）

评语:

修改意见:

中期考核专家小组签名：

组长_____

成员_____　_____　_____　_____

时间：　　年　　月　　日