

同行专家业内评价意见书编号： 20250854478

附件1

浙江工程师学院（浙江大学工程师学院） 同行专家业内评价意见书

姓名： 毛梓宇

学号： 22260275

申报工程师职称专业类别（领域）： 电子信息

浙江工程师学院（浙江大学工程师学院）制

2025年05月29日

填表说明

一、本报告中相关的技术或数据如涉及知识产权保护、军工项目保密等内容，请作脱密处理。

二、请用宋体小四字号撰写本报告，可另行附页或增加页数，A4纸双面打印。

三、表中所涉及的签名都必须用蓝、黑色墨水笔，亲笔签名或签字章，不可以打印代替。

四、同行专家业内评价意见书编号由工程师学院填写，编号规则为：年份4位+申报工程师职称专业类别(领域)4位+流水号3位，共11位。

一、个人申报

（一）基本情况【围绕《浙江工程师学院（浙江大学工程师学院）工程类专业学位研究生工程师职称评审参考指标》，结合该专业类别(领域)工程师职称评审相关标准，举例说明】

1. 对本专业基础理论知识和专业技术知识掌握情况(不少于200字)

在研究生阶段，我系统地学习了与工业系统安全相关的基础理论与核心技术，主要包括系统工程、安全工程、控制理论、嵌入式系统、网络安全等方面的内容。尤其在形式化方法方面，我重点掌握了模型检测、定理证明、时序逻辑等理论基础，并能够应用如Tamarin、Isabelle等工具对工业系统进行建模与验证。通过多个项目训练，我不仅加深了对形式化规范与验证方法的理解，也提升了将抽象模型转化为实际系统行为的能力。此外，我还掌握了Python、LaTeX等编程与文档工具，为后续的科研与工程工作打下了坚实基础。

2. 工程实践的经历(不少于200字)

我曾参与多个与工业系统相关的科研与工程项目，涵盖工业实时操作系统安全验证、安全协议自动建模以及去中心化协议形式验证等方向。

在某次与企业合作的项目中，我基于其提出的去中心化身份认证协议方案，设计并实现了形式化建模方案，用以验证协议是否符合关键的安全性质。

在另一个项目中，我利用现有的并发逻辑验证框架，对一个实时操作系统的内存管理和进程通信模块进行验证，保证了其关键安全性质的满足。

此外，我还参与构建了一个基于时序逻辑的智能合约交易攻击的实时监测工具的开发，有效结合了工程实践与学术研究。这些经历不仅锻炼了我在实际环境下分析与建模的能力，也让我更深刻地理解了工业系统的复杂性与安全需求。

3. 在实际工作中综合运用所学知识解决复杂工程问题的案例（不少于1000字）

在我从事工业系统安全研究的过程中，曾深入参与一个聚焦于安全协议形式化验证自动化的技术研究项目。该项目面向工业界一个实际问题——

如何高效准确地验证网络通信协议的安全性，尤其是避免由协议设计缺陷引发的安全漏洞。

该问题在工业控制网络、移动通信、银行系统、物联网等关键场景中广泛存在，并且一旦发生安全问题，后果可能极为严重。因此，对安全协议进行形式化建模与验证，是确保其在实际应用中可信可靠的重要手段。然而传统形式化验证工具虽然强大，但使用门槛高、自动化程度低，限制了其在工业界的广泛落地。

在调研中我发现，当前主流的安全协议验证工具，如 ProVerif 和 Tamarin

Prover，通常要求用户手动将自然语言的协议规范（如 RFC

文档）转化为形式化模型。这一过程高度依赖人工经验，不仅费时费力，而且容易因理解偏差导致建模错误。尤其是在实际工业协议日益复杂、文档繁杂且具备大量领域特定细节的背景下，这种“全靠人力”的方式难以满足规模化验证的需求。因此，如何让普通工程人员也能便捷地开展协议验证，是我所要面对的现实工程挑战。

为解决这一问题，我结合自己在自然语言处理、形式化方法以及工业安全协议方面的积累，尝试提出并实现一种基于大语言模型（LLM）辅助的安全协议符号模型自动生成方法。我的核心思想是构建一套从自然语言协议文档到可验证符号模型的自动转换流程，以此降低使用验证工具的门槛，提升形式化验证在工程实践中的可用性和效率。

具体地，该方法包含以下几个关键技术步骤。首先，我设计了一个基于组合范畴语法的语义解析框架，借助大语言模型（如GPT）对协议文档进行结构化语义分析。不同于传统基于规则的抽取方式，LLM

提供了对自然语言中上下文信息和专业术语的更强理解能力，从而能更准确地识别协议中的实体（如参与者、消息、密钥）、结构（如会话顺序）和属性（如认证目标）。解析结果被转换为一种中间表示语言，具有领域特定性，可读性强，且便于进一步处理。

接下来，我引入了静态分析与交互修复机制，对中间表示中的语义错误、冲突或遗漏进行检测与纠正。用户可以基于图形化界面进行交互确认或微调，使整个建模过程既具备智能性，也保留人机协同的可控性。然后，我开发了一个格式转换器，将中间表示转换为 Sapic+ 规范语言，这是一种兼容 Tamarin Prover 的协议建模语言。最后，通过编写编译器组件，我将 Sapic+ 描述进一步编译成 Tamarin 可接受的内部符号模型，并将生成过程中的语义变换进行了形式化证明，确保语义保真性。

通过这个项目，我不仅将所学的形式化验证方法、语言语义处理技术与系统安全理论进行了综合运用，还面对了工程实践中的诸多现实问题，如协议文档不规范、领域术语歧义、模型表达能力有限等。这些挑战使我深刻体会到，将学术理论转化为工程可用的工具，需要在算法、工程实现、人机交互等多个层面共同发力。最终，该方法为工程师提供了一种相对低门槛、高准确率的协议建模辅助工具，有望在工业界推广，提升协议设计的安全保障能力。


（二）取得的业绩（代表作）【限填3项，须提交证明原件（包括发表的论文、出版的著作、专利证书、获奖证书、科技项目立项文件或合同、企业证明等）供核实，并提供复印件一份】					
1. 公开成果代表作【论文发表、专利成果、软件著作权、标准规范与行业工法制定、著作编写、科技成果获奖、学位论文等】					
成果名称	成果类别 [含论文、授权专利（含发明专利申请）、软件著作权、标准、工法、著作、获奖、学位论文等]	发表时间/授权或申请时间等	刊物名称/专利授权或申请号等	本人排名/总人数	备注
LLM-aided Automatic Modelling for Security Protocol Verification	TOP期刊	2025年04月27日	47th International Conference on Software Engineering	1/5	EI会议收录
Quantitative Runtime Monitoring of Ethereum Transaction Attacks	TOP期刊	2024年04月28日	The ACM Web Conference 2025	2/7	EI会议收录
Applying Rely-Guarantee Reasoning on Concurrent Memory Management and Mailbox in μ C/OS-II	会议论文	2023年09月26日	28th International Conference on Formal Methods for Industrial Critical Systems	2/5	EI会议收录

2. 其他代表作【主持或参与的课题研究项目、科技成果应用转化推广、企业技术难题解决方案、自主研发设计的产品或样机、技术报告、设计图纸、软课题研究报告、可行性研究报告、规划设计方案、施工或调试报告、工程实验、技术培训教材、推动行业发展中发挥的作用及取得的经济社会效益等】

(三) 在校期间课程、专业实践训练及学位论文相关情况	
课程成绩情况	按课程学分核算的平均成绩： 84 分
专业实践训练时间及考核情况(具有三年及以上工作经历的不作要求)	累计时间： 1 年（要求1年及以上） 考核成绩： 83 分
本人承诺	
个人声明：本人上述所填资料均为真实有效，如有虚假，愿承担一切责任，特此声明！	
申报人签名： 毛梓宇	

20260215

二、日常表现考核评价及申报材料审核公示结果

日常表现考核评价	非定向生由德育导师考核评价、定向生由所在工作单位考核评价： <input checked="" type="checkbox"/> 优秀 <input type="checkbox"/> 良好 <input type="checkbox"/> 合格 <input type="checkbox"/> 不合格 德育导师/定向生所在工作单位分管领导签字（公章）：  年 月 日
申报材料审核公示	根据评审条件，工程师学院已对申报人员进行材料审核（学位课程成绩、专业实践训练时间及考核、学位论文、代表作等情况），并将符合要求的申报材料在学院网站公示不少于5个工作日，具体公示结果如下： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过（具体原因： 工程师学院教学管理办公室审核签字（公章）： 年 月 日

浙江大学研究生院
攻读硕士学位研究生成绩表

学号：22260275		姓名：毛梓宇		性别：男		学院：工程师学院			专业：电子信息			学制：2.5年			
毕业时最低应获：24.0学分				已获得：26.0学分+4.0学分（本科生课程）					入学年月：2022-09			毕业年月：			
学位证书号：				毕业证书号：					授予学位：						
学习时间		课程名称		备注	学分	成绩	课程性质	学习时间		课程名称		备注	学分	成绩	课程性质
2022-2023学年秋季学期		工业互联网系统安全前沿技术			2.0	94	专业选修课	2022-2023学年春夏学期		高阶工程认知实践			3.0	88	专业学位课
2022-2023学年秋季学期		工程技术创新前沿			1.5	88	专业学位课	2023-2024学年冬季学期		“四史”专题			1.0	88	公共选修课
2022-2023学年秋季学期		工业互联网安全系统工程			2.0	83	专业学位课	2024-2025学年春季学期		公司金融			1.0	94	公共学位课
2022-2023学年秋冬学期		工程伦理			2.0	71	公共学位课	2024-2025学年春季学期		研究生英语应用能力提升			2.0	84	公共学位课
2022-2023学年秋冬学期		研究生论文写作指导			1.0	87	专业学位课	2022-2023学年秋冬学期		体育训练与比赛（B类队-男子篮球）			1.0	100	本科生课
2022-2023学年冬季学期		新时代中国特色社会主义思想理论与实践			2.0	89	公共学位课	2022-2023学年春夏学期		体育训练与比赛（B类队-男子篮球）			1.0	100	本科生课
2022-2023学年秋冬学期		工业系统动态建模求解及优化			2.0	93	专业学位课	2023-2024学年秋冬学期		体育训练与比赛（B类队-男子篮球）			1.0	100	本科生课
2022-2023学年冬季学期		产业技术发展前沿			1.5	88	专业学位课	2023-2024学年春夏学期		体育训练与比赛（B类队-男子篮球）			1.0	100	本科生课
2022-2023学年春季学期		数学建模			2.0	60	专业选修课			硕士生读书报告			2.0	通过	
2022-2023学年春季学期		自然辩证法概论			1.0	77	公共学位课								

说明：1. 研究生课程按三种方法计分：百分制，两级制（通过、不通过），五级制（优、良、中、及格、不及格）。

2. 备注中“*”表示重修课程。

学院成绩校核章：

成绩校核人：张梦依

打印日期：2025-06-03



LLM-aided Automatic Modeling for Security Protocol Verification

Ziyu Mao¹, Jingyi Wang^{1*}, Jun Sun², Shengchao Qin³, Jiawen Xiong⁴

¹ Zhejiang University, Hangzhou, China

² Singapore Management University, Singapore

³ Xidian University, China

⁴ East China Normal University, China

Abstract—Symbolic protocol analysis serves as a pivotal technique for protocol design, security analysis, and the safeguarding of information assets. Several modern tools such as TAMARIN and PROVERIF have been proven successful in modeling and verifying real-world protocols, including complex protocols like TLS 1.3 and 5G AKA. However, developing formal models for protocol verification is a non-trivial task, which hinders the wide adoption of these powerful tools in practical protocol analysis.

In this work, we aim to bridge the gap by developing an automatic method for generating symbolic protocol models using Large Language Models (LLMs) from protocol descriptions in natural language document. Although LLMs are powerful in various code generation tasks, it is shown to be ineffective in generating symbolic models (according to our empirical study). Therefore, rather than applying LLMs naively, we carefully decompose the symbolic protocol modeling task into several stages so that a series of formal models are incrementally developed towards generating the final *correct* symbolic model. Specifically, we apply LLMs for semantic parsing, enable lightweight manual interaction for disambiguation, and develop algorithms to transform the intermediate models for final symbolic model generation. To ensure the correctness of the generated symbolic model, each stage is designed based on a formal execution model and the model transformations are proven sound. To the best of our knowledge, this is the first work aiming to generate symbolic models for protocol verification from natural language documents. We also introduce a benchmark for symbolic protocol model generation, with 18 real-world security protocol's text description and their corresponding symbolic models. We then demonstrate the potential of our tool, which successfully generated correct models of moderate scale in 10 out of 18 cases. Our tool is released at [1].

Index Terms—Automatic modeling, Symbolic analysis, LLMs

I. INTRODUCTION

It is notoriously hard to design and implement security protocols. Among the variety of methods for analyzing security protocols, symbolic methods play an important role in security protocol verification. Many protocol verifiers, such as TAMARIN [2] and PROVERIF [3], have been developed to formally analyze the correctness and security of a protocol, based on a symbolic model of the protocol. The effectiveness and usefulness of these tools are evidenced by the verification of large-scale real-world protocols such as TLS 1.3 [4], 5G AKA [5], and EMV payment [6], which have uncovered critical and subtle security vulnerabilities.

* Corresponding author.

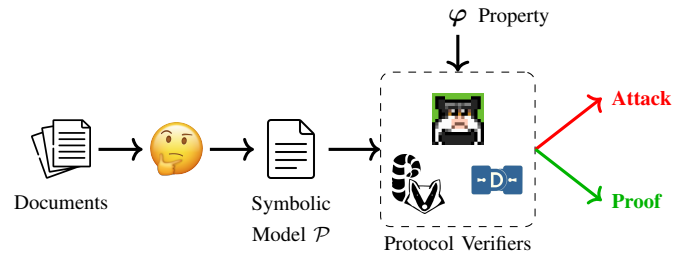


Fig. 1: Overall workflow for symbolic protocol analysis.

However, applying these tools in practice can be costly, as a user must first develop a formal symbolic model of the protocol, which requires not only expert knowledge about the protocol design, but also deep insights (as well as modeling skills) of the corresponding verifier. Take TAMARIN as an example. The overall workflow of protocol analysis is shown in Figure 1. To use TAMARIN, one must first develop a model of the security protocol in its input language (multiset rewriting rules). To do that, a user must first build, in the mind, an abstract model from the protocol document (e.g., IETF RFCs), and then encode the model using multiset rewriting rules which characterize a labeled transition system. Furthermore, the security properties φ to be verified must be encoded in first-order logic (FOL). The validity of verification results depends on whether the multiset rewriting rules and properties are modeled correctly, as well as, subtle choices made during the modeling, e.g., an untyped message in a rule may cause TAMARIN to treat their outgoing messages as input, leading to the non-termination of backwards reasoning [7]. To be fair, TAMARIN and other similar tools certainly made significant effort to make their modeling language accessible, and yet the difficulty in developing the symbolic model limits the application of these tools to ordinary users.

In this work, we aim to address this practical challenge by developing a method to generate symbolic models required by tools such as TAMARIN automatically. Note that there were previously some attempts with similar goals [8], [9], [10]. What makes this work different is that we integrate the capability of large language models (LLMs) into a multi-step approach and adopt a much more powerful intermediate modeling language. Note that although LLMs have shown remarkable capabilities in multiple ‘less-formal’ tasks like



Quantitative Runtime Monitoring of Ethereum Transaction Attacks

Xinyao Xu*
Zhejiang University
Hangzhou, China
xxinyao.Xu@zju.edu.cn

Ziyu Mao*
Zhejiang University
Hangzhou, China
maozyu@zju.edu.cn

Jianzhong Su
Sun Yat-Sen University
Zhuhai, China
sujzh3@mail2.sysu.edu.cn

Xingwei Lin
Zhejiang University
Hangzhou, China
xwlin.roy@gmail.com

David Basin
ETH Zurich
Zurich, Switzerland
basin@inf.ethz.ch

Jun Sun
Singapore Management University
Singapore, Singapore
junsun@smu.edu.sg

Jingyi Wang†
Zhejiang University
Hangzhou, China
wangjyee@zju.edu.cn

Abstract

The rapid growth of decentralized applications, while revolutionizing financial transactions, has created an attractive target for malicious attacks. Existing approaches to detecting attacks often rely on predefined rules or simplistic and overly-specialized models, which lack the flexibility to handle the wide spectrum of diverse and dynamically changing attack types.

To address this challenge, we present a general and extensible framework, MoE (**M**onitoring **E**thereum), that leverages *runtime verification* to detect a wide range of attacks on Ethereum. MoE features an expressive attack modeling language, based on Metric First-order Temporal Logic (MFOTL), that can formalize a wide range of attacks. We integrate a novel semantic lifting approach that extracts system behaviors relevant for various attacks, which can be analyzed using the monitoring tool MONPOLY. Furthermore, we also equip MoE with quantitative capabilities to evaluate the similarity between a transaction and an attack formula to enhance its performance in identifying attacks, including near-miss attacks.

We carry out extensive experiments with MoE on a labeled benchmark and a large-scale dataset containing over one million transactions. On the labeled benchmark, MoE successfully detects 92.0% attacks and achieves a 45.0% higher recall rate than competing state-of-the-art tool. MoE finds 3,319 attacks with 95.4% precision on the large dataset. Furthermore, MoE uses quantitative analysis to uncover 8% additional attacks. Finally, the average time for

monitoring a transaction is less than 23 ms, positioning MoE as a promising practical solution for real-time attack detection for Ethereum.

CCS Concepts

• Security and privacy → Web application security.

Keywords

Ethereum, Runtime Monitoring, Ethereum Attack Detection

ACM Reference Format:

Xinyao Xu, Ziyu Mao, Jianzhong Su, Xingwei Lin, David Basin, Jun Sun, and Jingyi Wang. 2025. Quantitative Runtime Monitoring of Ethereum Transaction Attacks. In *Proceedings of the ACM Web Conference 2025 (WWW '25)*, April 28-May 2, 2025, Sydney, NSW, Australia. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3696410.3714682>

1 Introduction

In the realm of blockchain and smart contract technologies, the decentralized application (DApp) ecosystem has gained substantial attention [22–24]. Smart contracts are now widely used, in particular in the financial sector [9], and manage a wide range of assets [18]. Ethereum, the driving force behind these applications, has witnessed a remarkable increase in its market capitalization [1]. Such milestones underscore the vast potential of this ecosystem, marking it a key area of development in the world of digital finance.

Unfortunately, this surge has also brought forth a darker reality: transaction-level attacks, which result in illegal financial gains on Ethereum, are becoming a trend [18]. The ValueDeFi incident[13] exemplifies the severe impact of such attacks, where an attacker exploited the MultiStables library via a flash loan, causing a loss of 6 million USD [13]. Recently, innovative transaction-level attacks such as call injection and sandwich attacks have emerged, undermining transaction integrity and manipulating market outcomes [4, 25]. Specifically, call injection attacks perturb smart contract operations by altering function calls, resulting in unauthorized

*Xinyao Xu and Ziyu Mao contributed equally.

†Jingyi Wang is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '25, April 28-May 2, 2025, Sydney, NSW, Australia

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1274-6/25/04

<https://doi.org/10.1145/3696410.3714682>



Applying Rely-Guarantee Reasoning on Concurrent Memory Management and Mailbox in $\mu\text{C}/\text{OS-II}$: A Case Study

Huan Sun, Ziyu Mao, Jingyi Wang^(✉), Ziyang Zhao, and Wenhai Wang^(✉)

Zhejiang University, Hangzhou, China

{huansun,maozyu,wangjyee,zhaoziyan,zdzzlab}@zju.edu.cn

Abstract. Real-time operating systems (RTOSs) such as $\mu\text{C}/\text{OS-II}$ are critical components of many industrial systems, which makes it of vital importance to verify their correctness. However, earlier specifications for verification of RTOSs often do not explicitly specify the behavior of possible unbounded kernel service invocations. To address the problem, a new event-based modelling approach is recently proposed to treat the operating system as a concurrent reactive system (CRS). Besides, a respective parametric rely-guarantee style reasoning framework called PiCore is developed to verify such systems effectively. Witnessing the advancement, we conduct a case study to investigate the use of PiCore to compositionally verify two important entangled modules of a practical RTOS $\mu\text{C}/\text{OS-II}$, i.e., the memory management module and the mailbox module. Several desirable safety properties regarding the memory pools and mailboxes are formally defined and proved with PiCore (≈ 2500 lines of specifications and proof scripts in Isabelle/HOL) based on a formal execution model considering the two modules simultaneously. We also discuss the shortcomings of PiCore for our case study and present possible improvement directions.

1 Introduction

Real-time operating systems (RTOSs) are the fundamental basis to support real-time applications. They provide a framework for effectively managing the execution of tasks and interrupts, ensuring that critical tasks are given sufficient priority so that timing constraints are met. This is achieved by the cooperation of several important mechanisms integrated within an RTOS: 1) scheduling mechanisms that determine whether and how tasks should execute, 2) mechanisms for managing system resources such as memory and processing power, and 3) inter-process communication (IPC) mechanisms that enable different processes or threads to exchange information and synchronize their activities.

Nowadays, RTOSs are increasingly adopted in various safety-critical industrial control systems, including aerospace, automotive, and medical devices. Every single bug or vulnerability presented in the RTOSs can have far-reaching