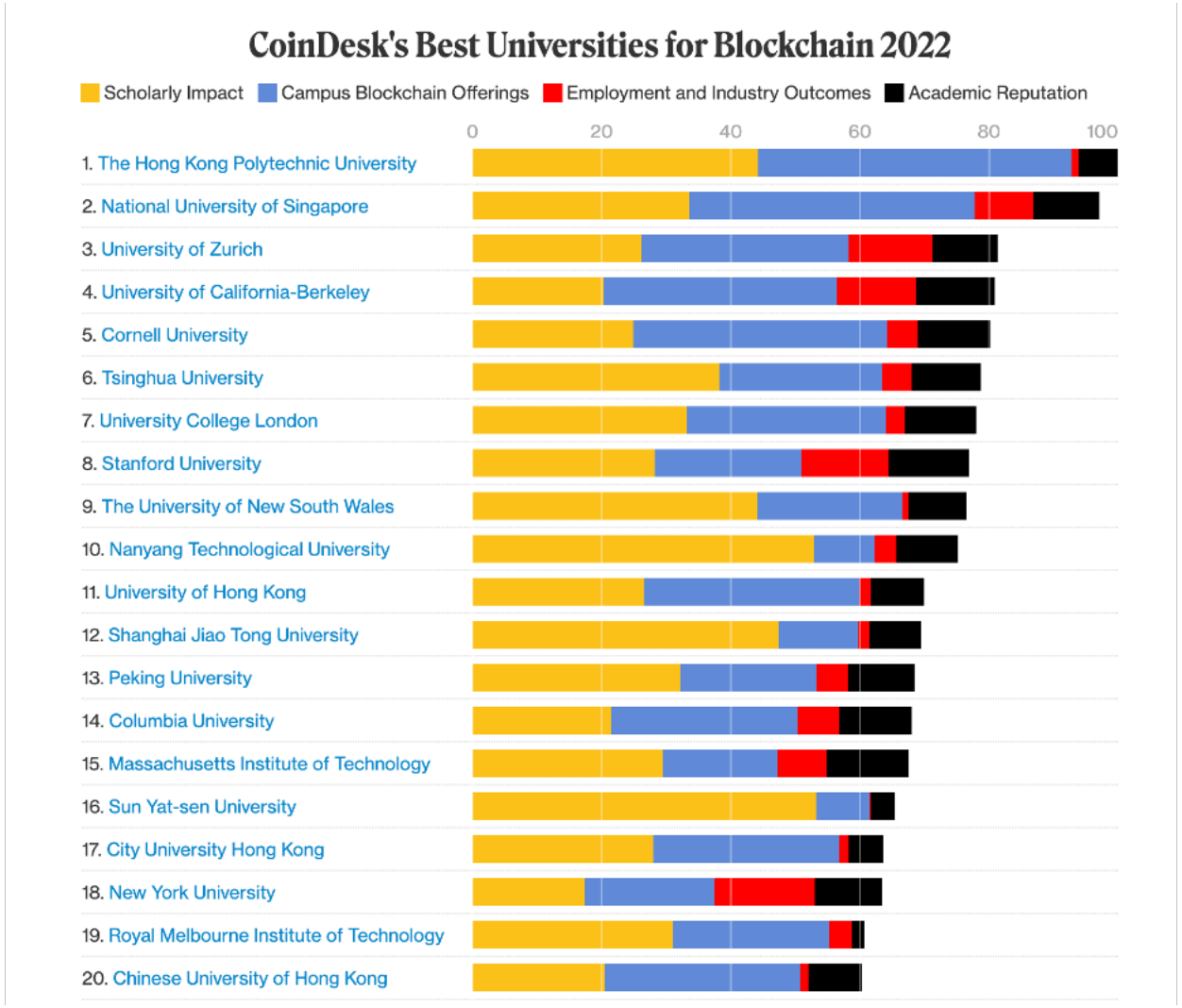
Blockchain Room of Requirement (BR^2): An LLM-Enhanced Simulator for Blockchain Protocols

Cong Wang
Professor
Department of Computer Science
City University of Hong Kong

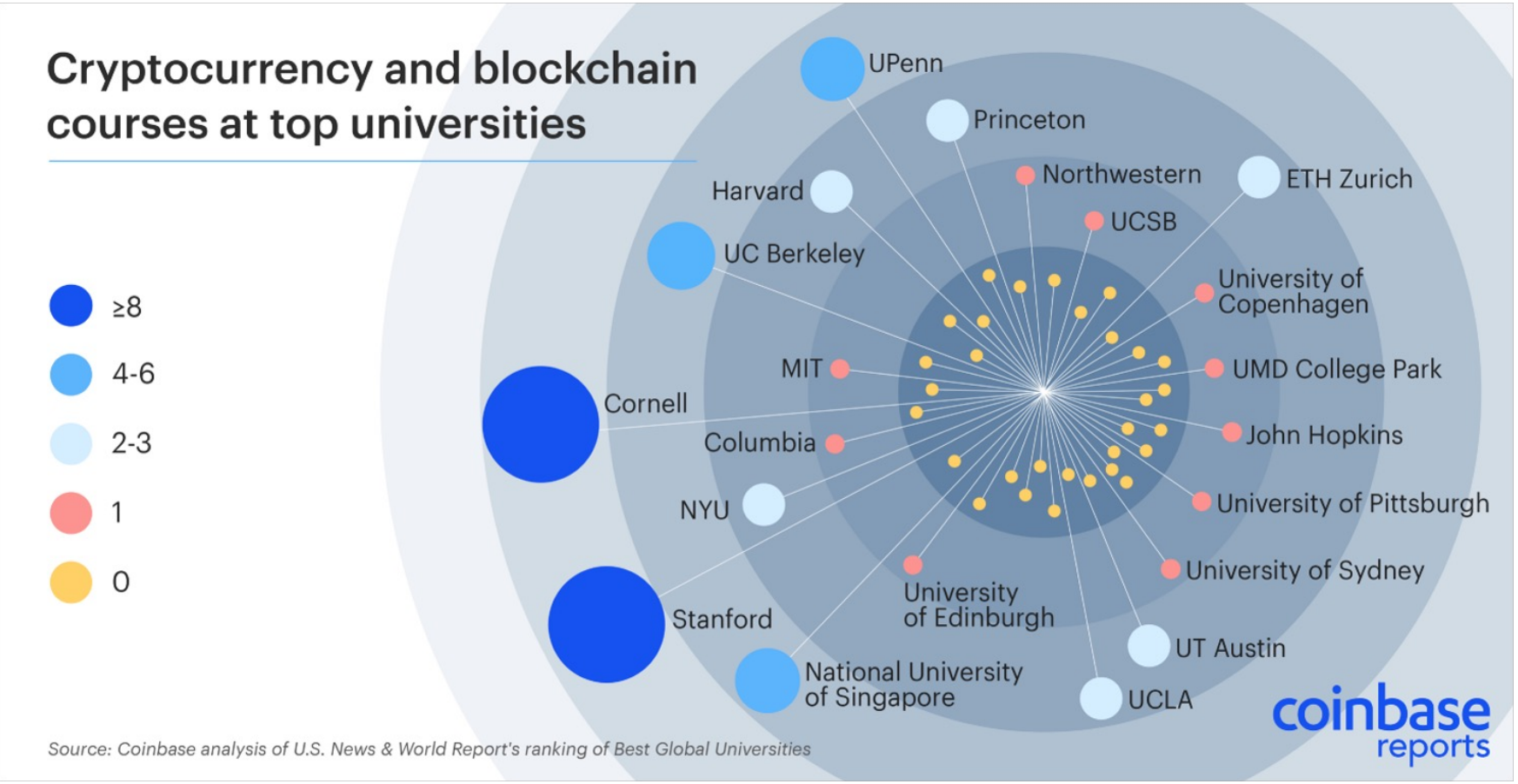
July 9, 2024

Blockchain Education



CoinDesk. Best Universities for Blockchain 2022. https://www.coindesk.com/layer2/2022/09/26/best-universities-for-blockchain-2022/

Blockchain Education



Coinbase. The rise of crypto in higher education. https://www.coinbase.com/blog/the-rise-of-crypto-in-higher-education

The Teaching Issue

Students feel confused after blockchain classes…

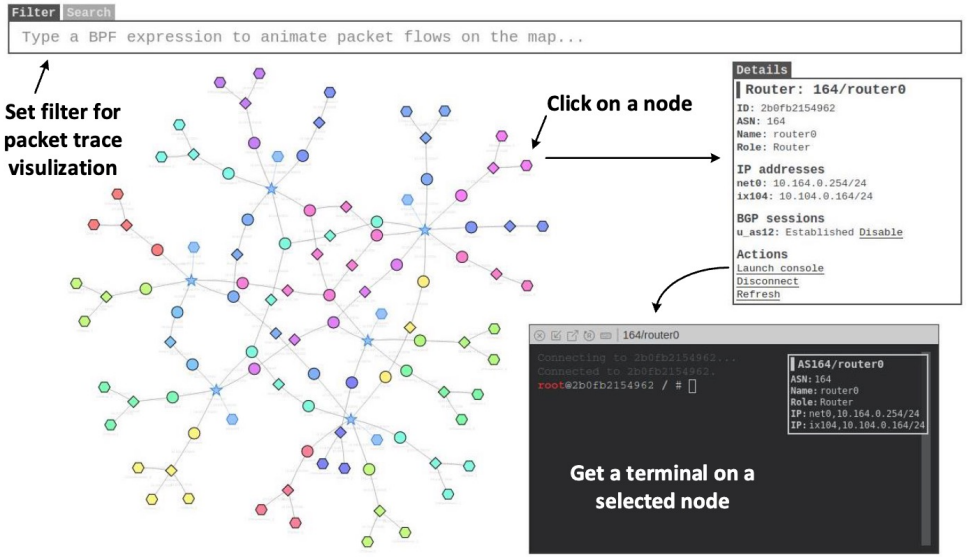
- How do cryptographic hashes and digital signatures work?
- What are the effects of decreasing Bitcoin's block interval?
- How can one get hands-on experience reproducing security incidents?
- Why do some “improvement” solutions fail?
- How do the economic models behind protocols function?
- How can security risks beyond the code level be recognized?
- ……

The Teaching Issue

Similar issue also happens in Internet education

The *SEED Emulator* [HotNets'22]

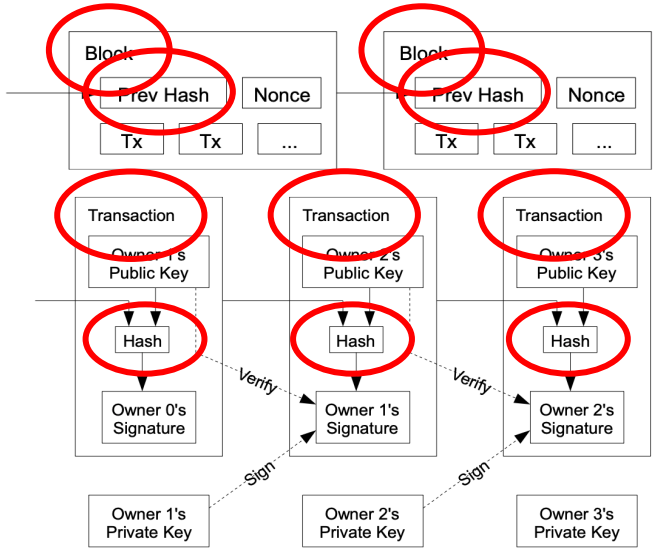
So we seek for some playgrounds and tools of blockchains



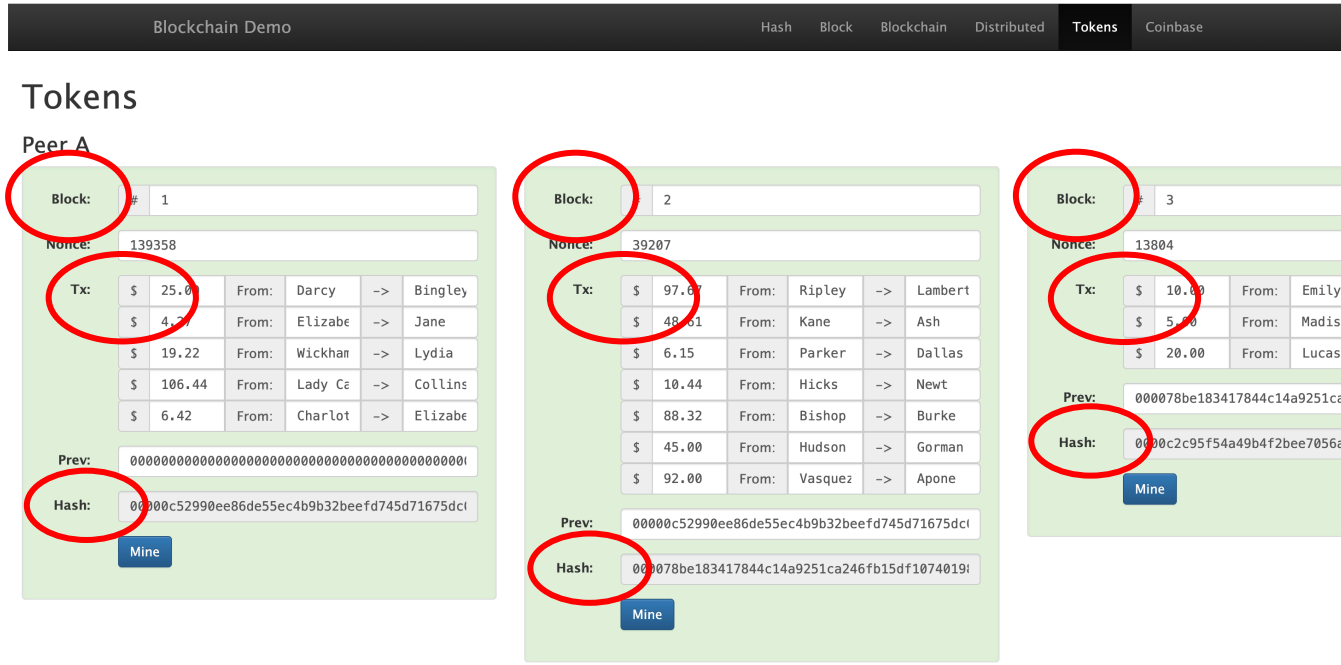
Wenliang Du, Honghao Zeng, Kyungrok Won. 2022. SEED Emulator: An Internet Emulator for Research and Education. In Proc. of HotNets.

The Blockchain Demo

A web-based demonstration: blocks, transactions, hashes, etc.



Original Bitcoin architecture

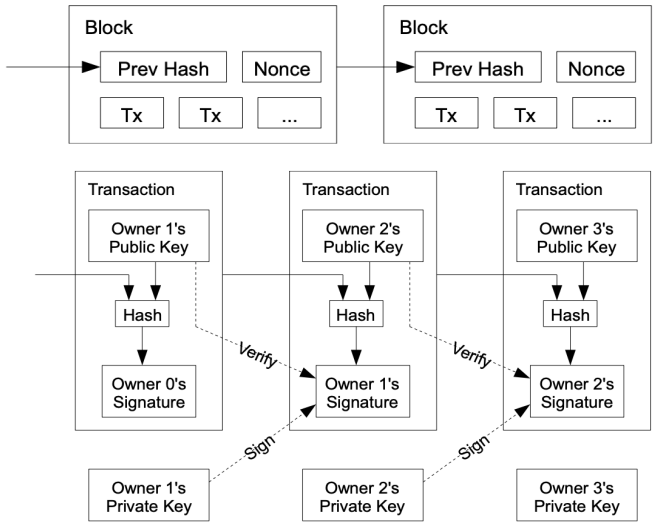


Simply a distributed ledger

Anders Brownworth. Blockchain Demo. https://andersbrownworth.com/blockchain/

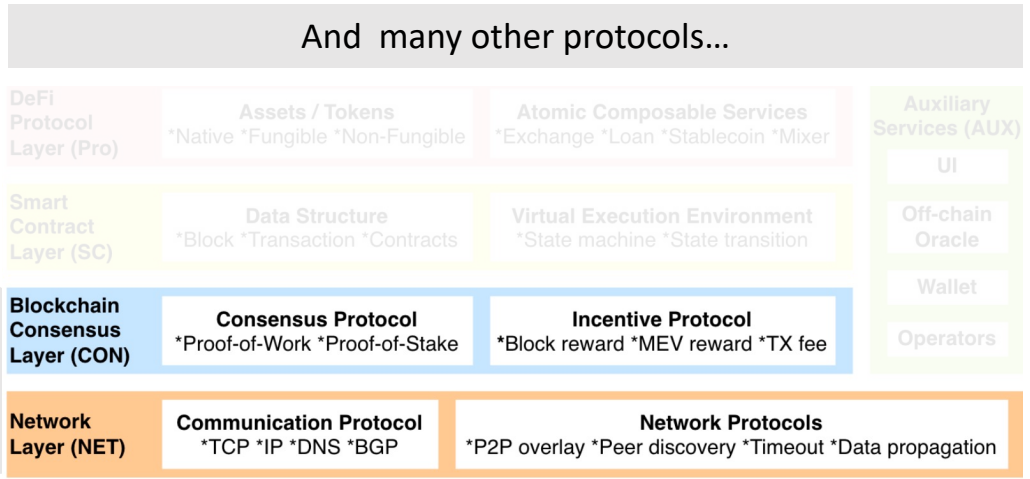
Blockchain Evolves…

Distributed ledger



Original Bitcoin architecture

Decentralized computing platform



Current layered blockchain architecture [S&P'23]

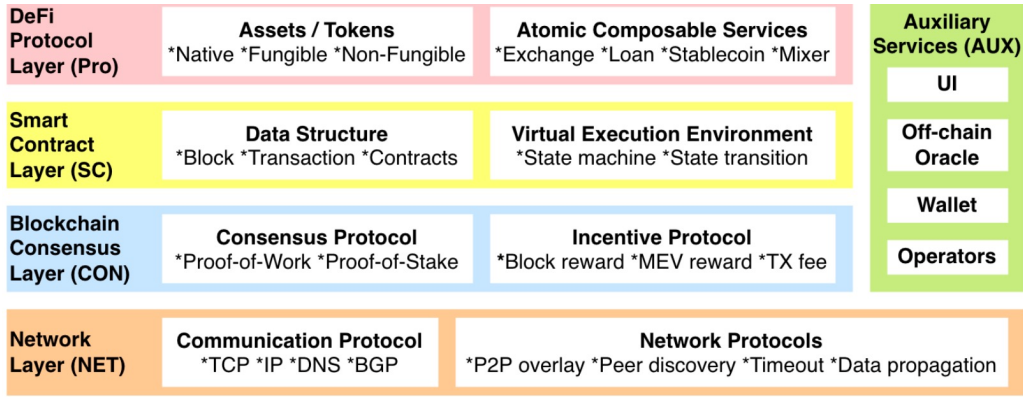
Satoshi Nakamoto, 2008, Bitcoin: A Peer-to-Peer Electronic Cash System. Online at: https://bitcoin.org/bitcoin.pdf.
Liyi Zhou et al. 2023. SoK: Decentralized Finance (DeFi) Attacks. In Proc. of IEEE S&P.

Rapid Evolution of Blockchain

Opportunities 😀

- Market cap increase
- Scalability improvements
- Smart contract evolution

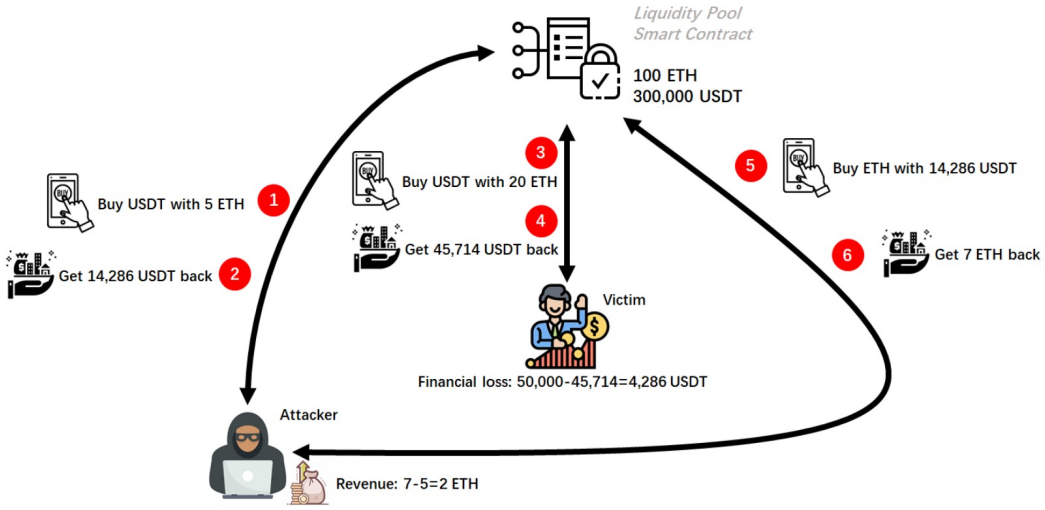
...



Current layered blockchain architecture [S&P'23]

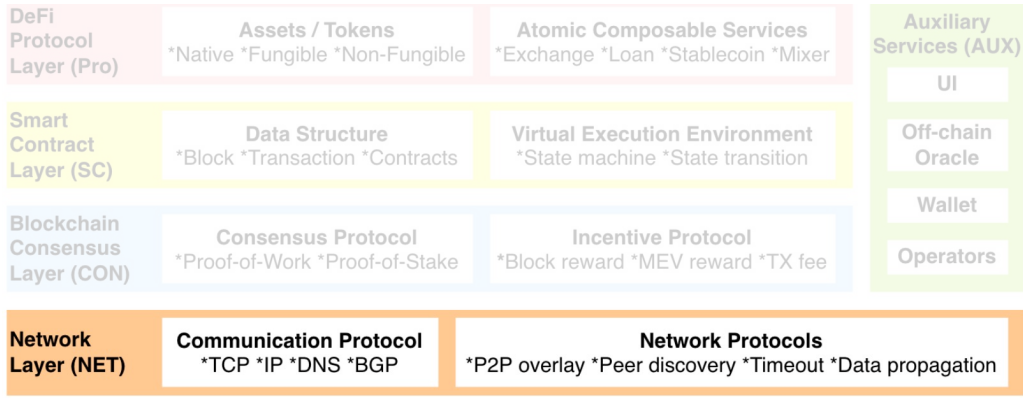
Liyi Zhou et al. 2023. SoK: Decentralized Finance (DeFi) Attacks. In Proc. of IEEE S&P.
Global Live Cryptocurrency Charts & Market Data. https://coinmarketcap.com/charts/

Rapid Evolution of Blockchain



Sandwich attack [CHI'22]

Broadened attack surfaces 🤯

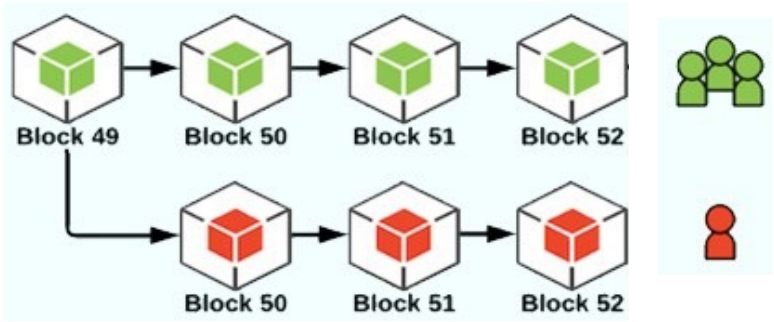


Current layered blockchain architecture [S&P'23]

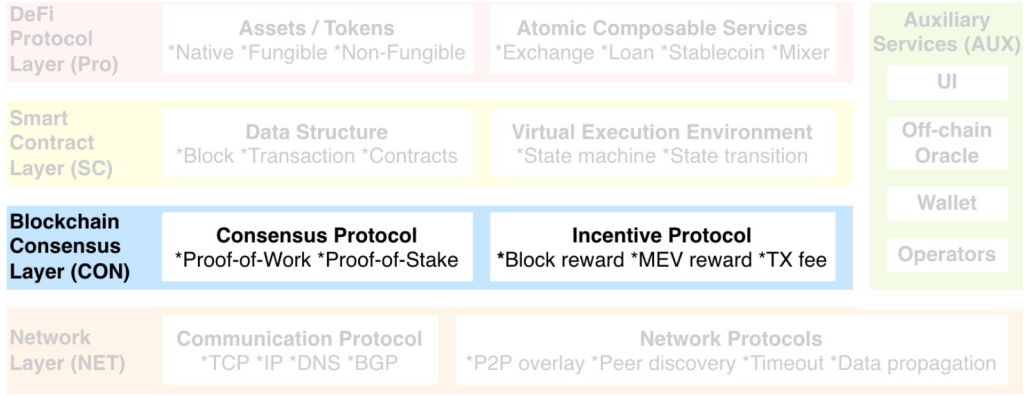
Ye Wang, Patrick Zuest, Yaxing Yao, et al. 2022. Impact and User Perception of Sandwich Attacks in the DeFi Ecosystem. In Proc. of ACM CHI.
Liyi Zhou et al. 2023. SoK: Decentralized Finance (DeFi) Attacks. In Proc. of IEEE S&P.

Rapid Evolution of Blockchain

Broadened attack surfaces 🤯



51% attack



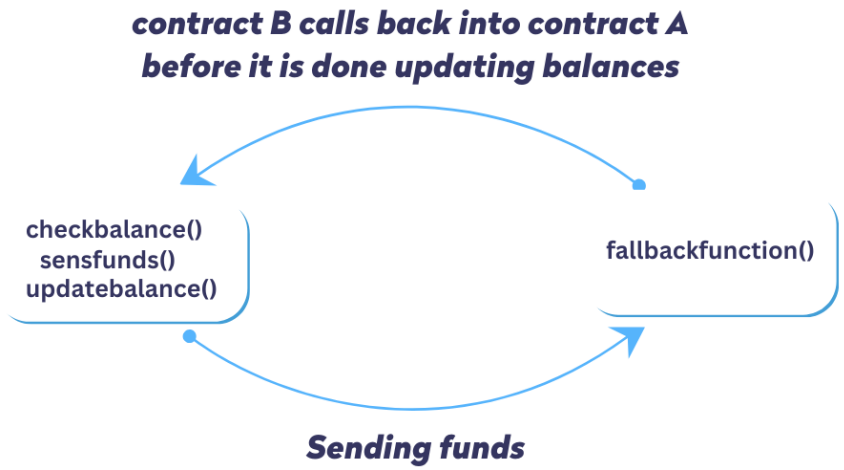
Current layered blockchain architecture [S&P'23]

List of Blockchain Attack Vectors and Vulnerabilities You Should Know, https://kingslanduniversity.com/blockchain-attack-vectors-vulnerabilities

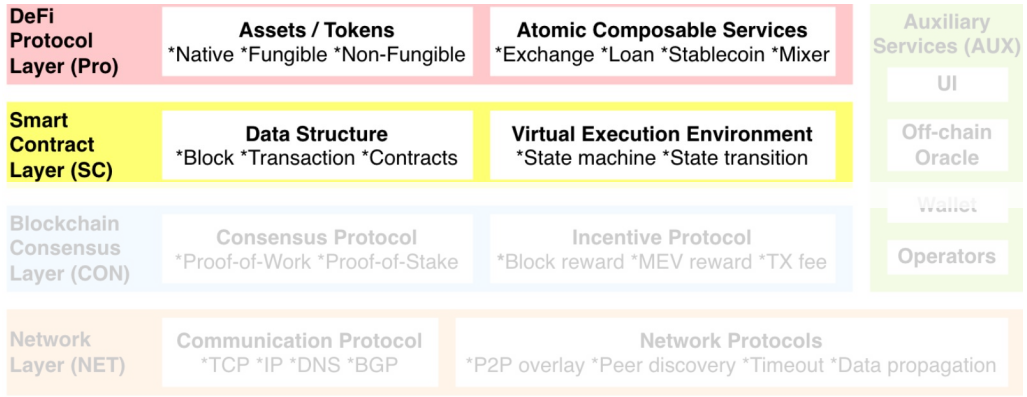
Liyi Zhou et al. 2023. SoK: Decentralized Finance (DeFi) Attacks. In Proc. of IEEE S&P.

Rapid Evolution of Blockchain

Broadened attack surfaces 🤯



Reentrancy attack



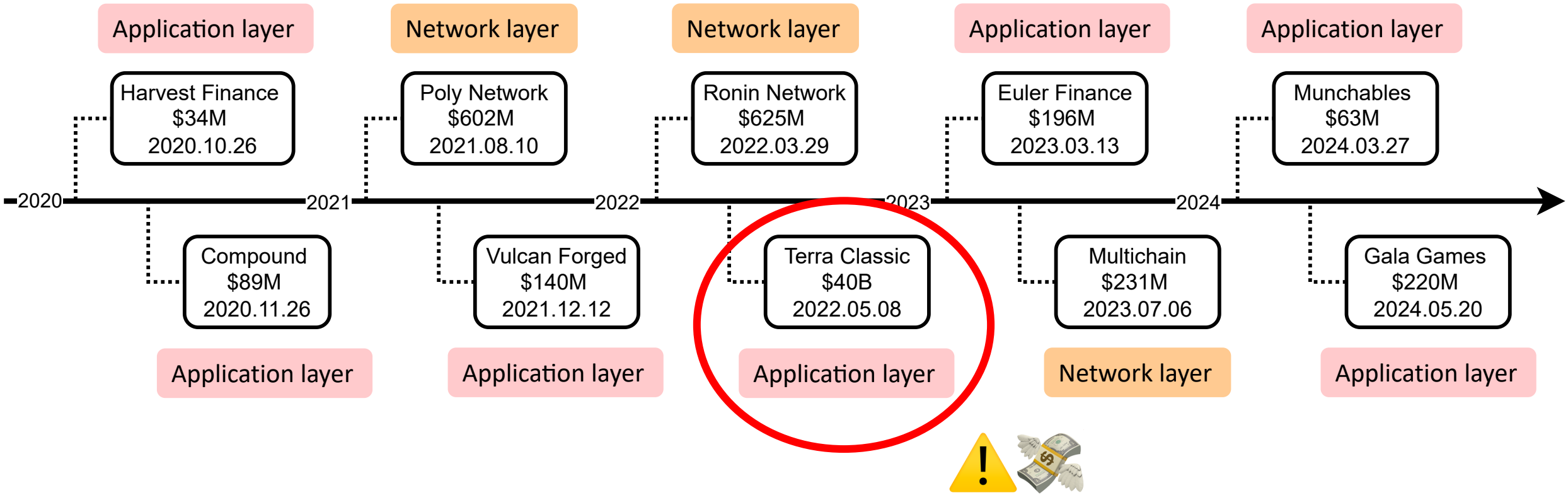
Current layered blockchain architecture [S&P'23]

What is a Reentrancy Attack in Smart Contracts and How to Prevent It? https://medium.com/blockchain-hacks/what-is-reentrancy-attack-in-smart-contracts-and-how-to-prevent-them-d65ad76dce5f

Liyi Zhou et al. 2023. SoK: Decentralized Finance (DeFi) Attacks. In Proc. of IEEE S&P.

Rapid Evolution of Blockchain

Not isolated cases; such issues keep occurring…



Three pieces of advice on how to learn computer security:

- **Study. Studying** can take many forms. It can be classwork…reading…
- **Do.** Computer security is fundamentally a **practitioner's art**, and that requires practice...
- **Show.** It doesn't matter what you know or what you can do if you can't **demonstrate** it to someone...

Bruce Schneier

So You Want to Be a Security Expert? https://www.schneier.com/blog/archives/2012/07/how_to_become_a_1.html

Three pieces of advice on how to learn computer security:

- **Study. Studying** can take many forms. It can be classwork…reading…
- **Do.** Computer security is fundamentally a **practitioner's art**, and that requires practice...
- **Show.** It doesn't matter what you know or what you can do if you can't **demonstrate** it to someone...

So You Want to Be a Security Expert? https://www.schneier.com/blog/archives/2012/07/how_to_become_a_1.html

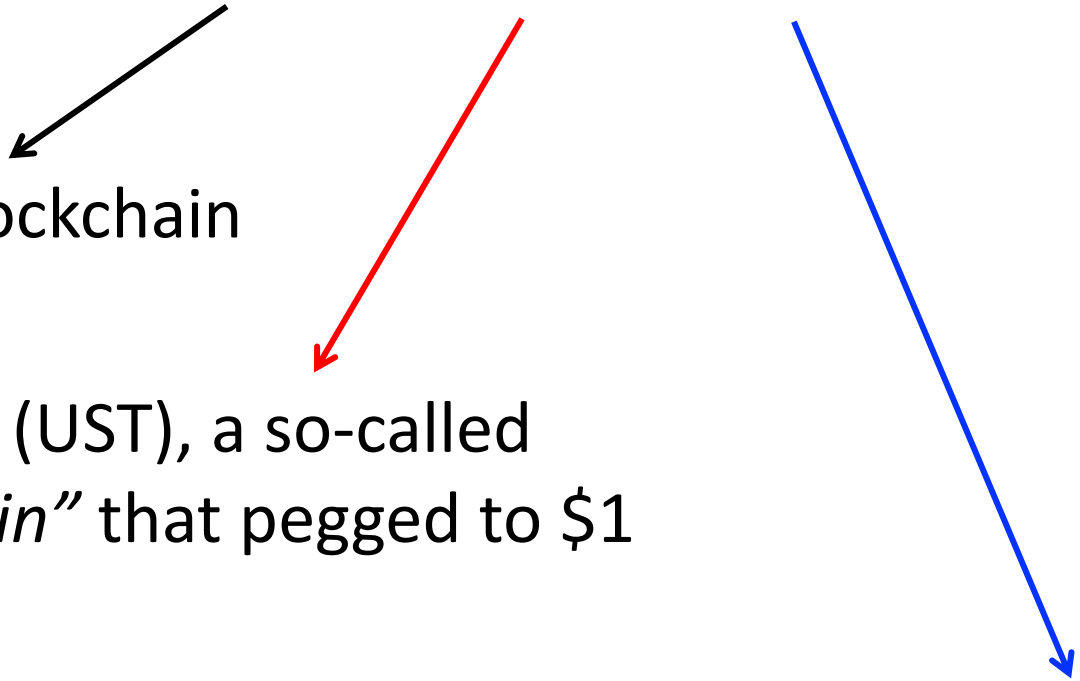
We need a place for

- ***Getting hands-on experience (DO)****:* reproduce incidents that

Actually we already have a few places to ***DO*** and ***SHOW***

- ***Demonstration (SHOW)****:* uncover potential consequences (e.g., security issues) before deployment

Case Study: Terra UST/LUNA Incident



A new blockchain

TerraUSD (UST), a so-called *"stablecoin"* that pegged to $1

LUNA token to help stabilize UST via ***arbitrage***:

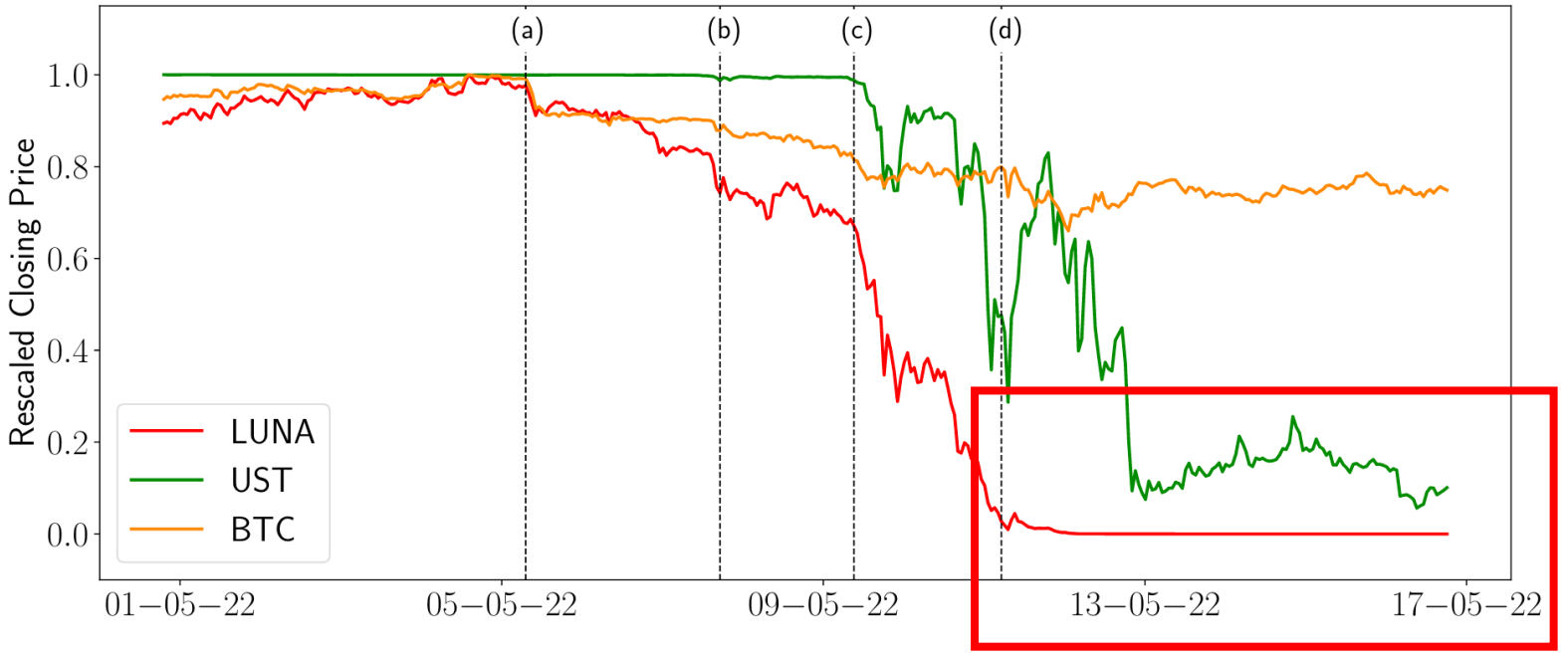
- A fixed exchange mechanism: 1 UST <=> $1 worth of LUNA
- If UST $0.9: buy 1 UST => exchange for LUNA => sell for $1 (earn $0.1) => UST up
- If UST $1.1: buy $1 LUNA => exchange for UST => sell for $1.1 (earn $0.1) => UST down

Terra Docs. https://docs.terra.money/learn/protocol

Case Study: Terra UST/LUNA Incident

The “stablecoin” seems not quite “stable”

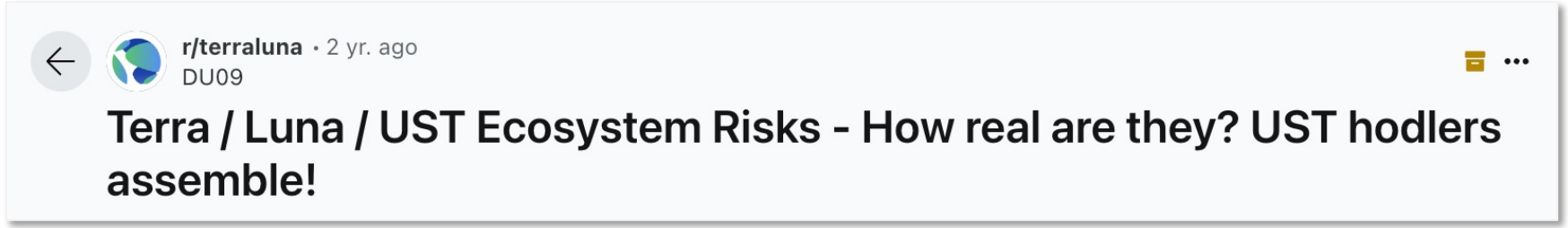
- LUNA crash and UST depeg (far from $1)

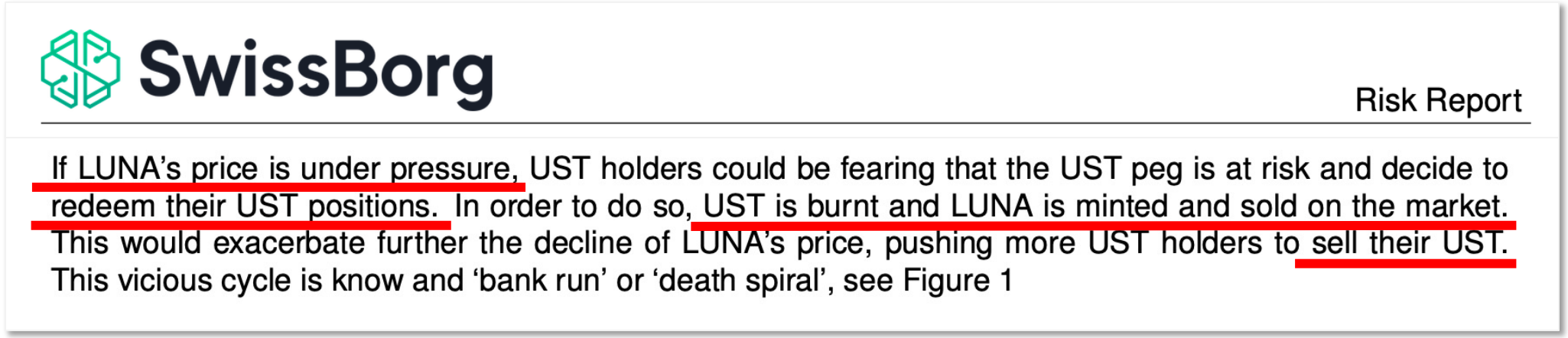


Almost $0!

Antonio Briola, David Vidal-Tomas, Yuanrong Wang, and Tomaso Aste. 2023. Anatomy of a Stablecoin’s failure: the Terra-Luna case. Finance Research Letters.

Case Study: Terra UST/LUNA Incident





How to demonstrate this? 🤔

DU09. 2022. Terra / Luna / UST Ecosystem Risks - How real are they? UST hodlers assemble!. https://www.reddit.com/r/terraluna/comments/s2bnbw/terra_luna_ust_ecosystem_risks_how_real_are_they/
SwissBorg. TerraLuna & UST - Risk Assessment. https://app.hubspot.com/documents/7219152/view/296921981?accessId=4d1141

Case Study: Terra UST/LUNA Incident

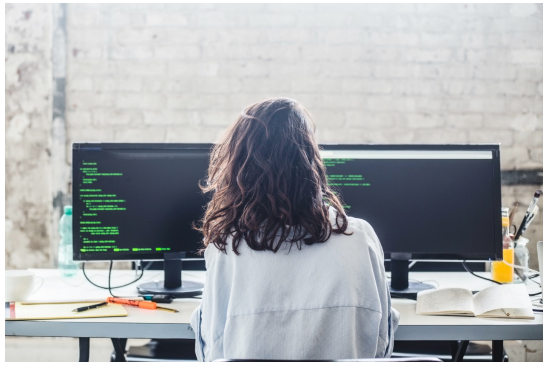
Replay historical transactions?

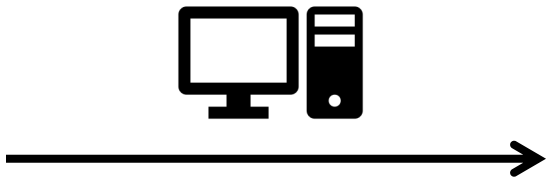
Yes, but what if we want to try more?

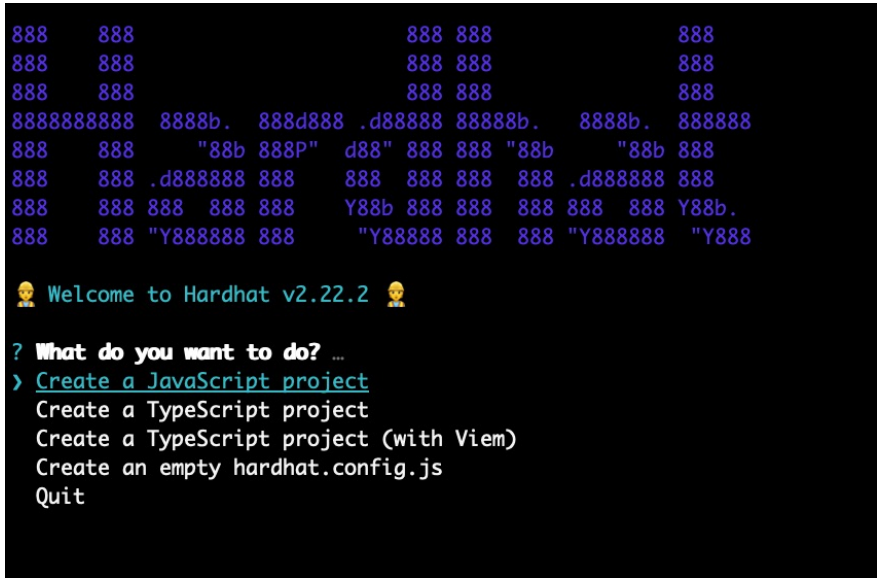
How to “poke around” with it? 🤔

Starting Point: Local Simulator  Hardhat

- A blockchain environment that runs on the ***local machine*** and allows for *quick* development and testing,
- e.g., Ethereum and Hardhat







Ethereum development environment for professionals, https://hardhat.org/

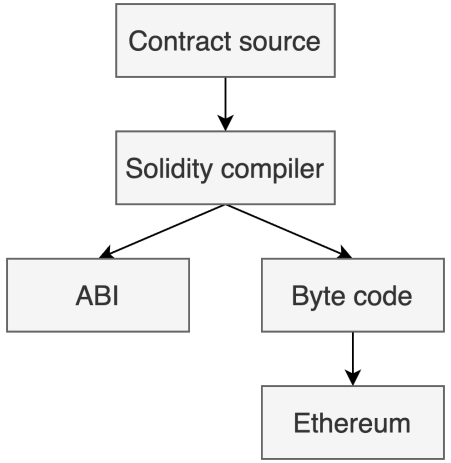
Starting Point: Local Simulator Hardhat

A dominant blockchain simulator

- supported by Ethereum Foundation
- common usage, strong community support, and extensive documentation and plugins

Documentation, https://hardhat.org/docs

Starting Point: Local Simulator Hardhat

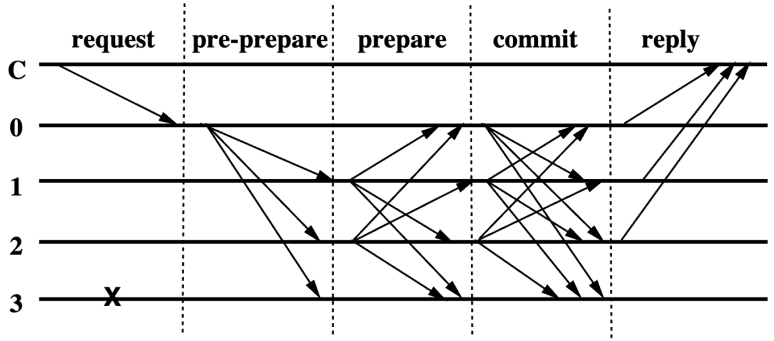


Inclusions:

- Local network with single node
- Smart contract compilation and deployment
- Transaction execution

Exclusions:

- Consensus mechanism
- Network synchronization
- Block mining



Ethereum development environment for professionals, https://hardhat.org/docs
Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In Proc. of USENIX OSDI.
Coinbase. What is mining? https://www.coinbase.com/en-sg/learn/crypto-basics/what-is-mining

Basic Workflow on  Hardhat

Installation & Initialization

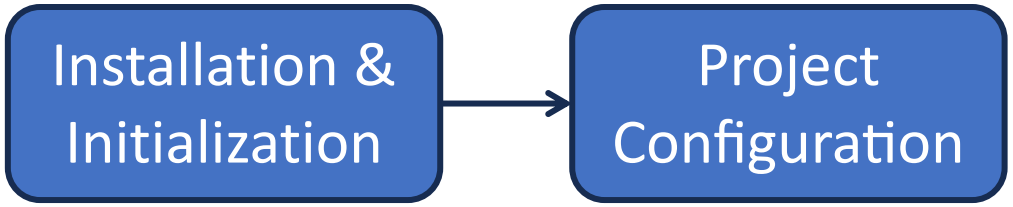
```
sudo apt update
sudo apt install curl git
curl -fsSL https://deb.nodesource.com/setup_22.x | sud
bash –
sudo apt-get install -y nodejs
npm init
npm install --save-dev hardhat
npx hardhat init
```

```
➜  hardhat-tutorial npx hardhat init

888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888


👷 Welcome to Hardhat v2.22.5 👷


? What do you want to do? …
  Create a JavaScript project
  Create a TypeScript project
  Create a TypeScript project (with Viem)
❯ Create an empty hardhat.config.js
  Quit
```

all you need!

https://hardhat.org/docs

Basic Workflow on Hardhat




```
➜  hardhat-tutorial npx hardhat init

888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888


👷 Welcome to Hardhat v2.22.5 👷


? What do you want to do? …
  Create a JavaScript project
  Create a TypeScript project
  Create a TypeScript project (with Viem)
❯ Create an empty hardhat.config.js
  Quit
```

https://hardhat.org/docs

The *hardhat.config.js* file

~~Copy and paste?~~ (off-the-shelf configuration is rarely the best fit)

Have to go through the learning curve:

i. Understand the configuration options
   - Read the software document
   - Get familiar with libraries & protocols

ii. Modify the configuration file

iii. Check the correctness

```
require("@nomicfoundation/hardh
at-toolbox");
/** @type
import('hardhat/config')... */ }
module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    hardhat: {
      chainId: 31337,
      from: 0x123,
      gas: 8000000,
      gasPrice: 20000000000,
      gasMultiplier: 1,
      minGasPrice: 0,
      initialBaseFeePerGas:
"100000",
      accounts: {
        mnemonic: "test
test ...",
 accountsBalance: "10000000",
        ...
        count: 20,
},
      [
        {
          privateKey: "PK1",
          balance:
"1000000000000",
        },
        ...
      ],
      blockGasLimit:
30_000_000,

allowUnlimitedContractSize:
false,

allowBlocksWithSameTimestamp:
false,
      forking: {
        url:
"https://mainnet.infura.io/..."
,
        blockNumber: 12345678,
        ...
      },
      hardfork: "shanghai",
      chains: {
        1: {
          hardforkHistory: {
    ...
  },
  ...
```

Hardhat network

Gas

Mining

Accounts

JSON-RPC networks

Blocks

Solidity

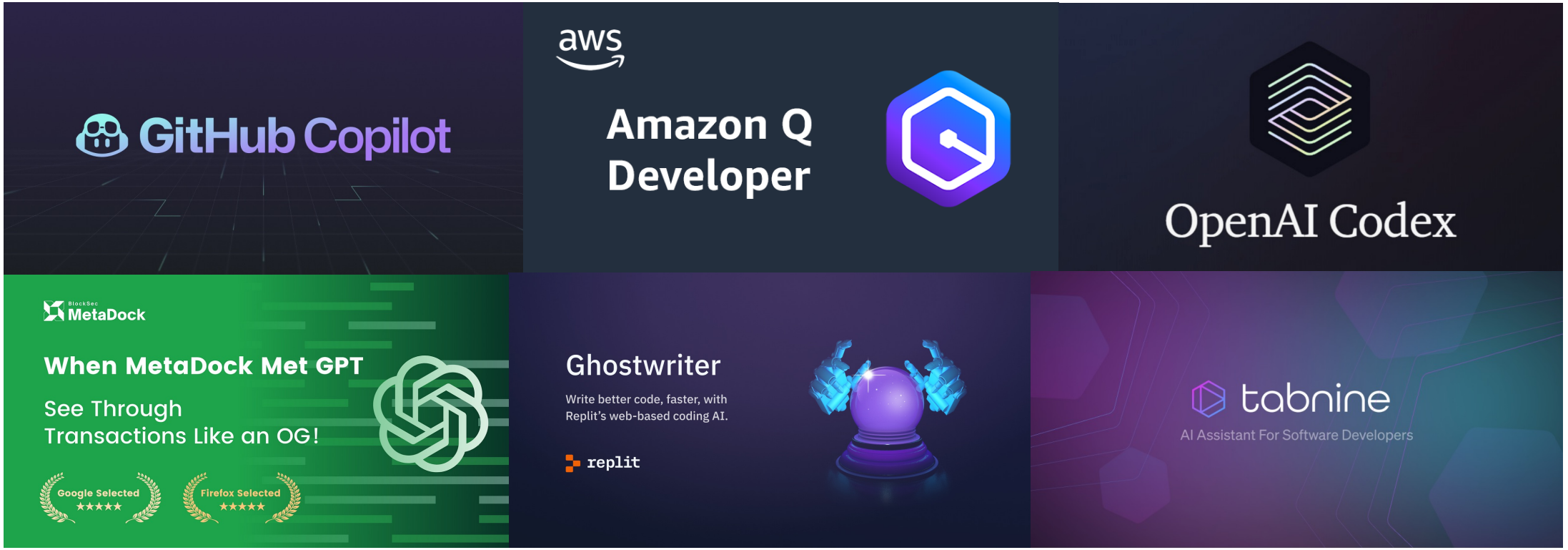
EVM

Hardforks

...

The Trend: ChatGPTed Interface

The LLM trend where many applications are being “*ChatGPTed*”



The *hardhat.config.js* file

~~Copy and paste?~~ (off-the-shelf configuration is rarely the best fit)

Have to go through the learning curve:

i. Understand the configuration options
   - Read the software document
   - Get familiar with libraries & protocols

ii. Modify the configuration. file

Also "ChatGPTed?"

iii. Check the correctness

```
require("@nomicfoundation/hardh    ...
at-toolbox");                    },
/** @type                       ...
import('hardhat/config')... */ }
module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    hardhat: {
      chainId: 31337,
      from: 0x123,
      gas: 8000000,
      gasPrice: 20000000000,
      gasMultiplier: 1,
      minGasPrice: 0,
      initialBaseFeePerGas:
"100000",
      accounts: {
        mnemonic: "test
test ...",
 accountsBalance: "10000000",
        ...
         count: 20,
},
      [
        {
          privateKey: "PK1",
          balance:
"1000000000000",
        },
        ...
      ],
      blockGasLimit:
30_000_000,

allowUnlimitedContractSize:
false,

allowBlocksWithSameTimestamp:
false,
      forking: {
        url:
"https://mainnet.infura.io/..."
,
        blockNumber: 12345678,
        ...
      },
      hardfork: "shanghai",
      chains: {
        1: {
          hardforkHistory: {
```

LLM-assisted Structured Output Generation

Natural Language

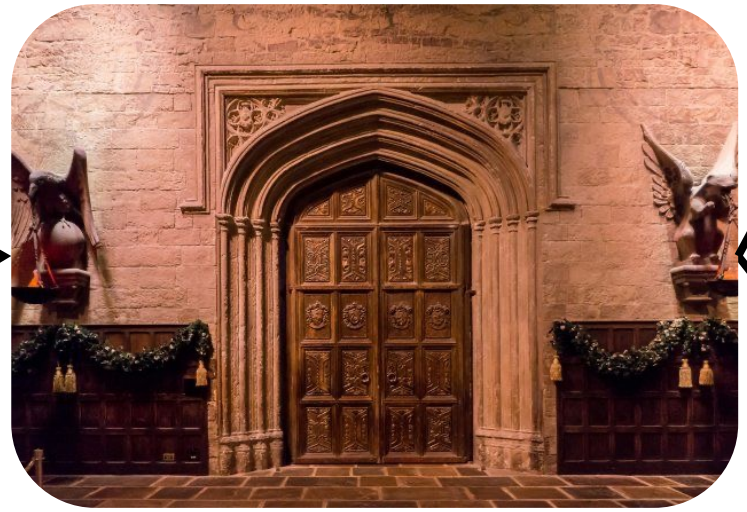
Structured Output (specification, script, …)

- *Network configuration* [NeurIPS'22, HotNets'23, CoNEXT'24]
- *Software specification* [HotNets'23]
- *Test script generation* [QRS'23, TSE'24]
- …

Luca Beurer-Kellner, Martin Vechev, Laurent Vanbever, et al. 2022. Learning to Configure Computer Networks with Neural Algorithmic Reasoning. In Proc. of NeurIPS.
Sathiya Kumaran Mani, Yajie Zhou, Kevin Hsieh, et al. 2023. Enhancing Network Management Using Code Generated by Large Language Models. In Proc. of HotNets.
Changjie Wang, Mariano Scazzariello, Alireza Farshin, et al. 2024. NetConfEval: Can LLMs Facilitate Network Configuration? In Proc. of CoNEXT.
Prakhar Sharma and Vinod Yegneswaran. 2023. PROSPER: Extracting Protocol Specifications Using Large Language Models. In Proc. of HotNets.
Shengcheng Yu, Chunrong Fang, Yuchen Ling, et al. 2023. LLM for Test Script Generation and Migration: Challenges, Capabilities, and Opportunities. In Proc. of QRS.
Max Schäfer, Sarah Nadi, Aryaz Eghbali, et al. 2024. An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation. TSE.

Room of Requirement

Harry Potter



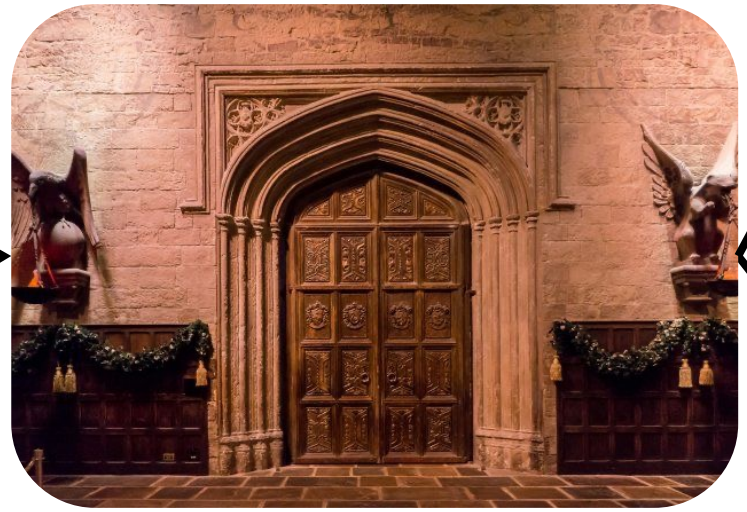
The magical room that transforms to meet the seeker's needs

Configured Environment

Intended Items

Room of Requirement

in Blockchain: BR^2



The magical room that transforms to meet the seeker's needs

Streamlined Custom Configuration

Intended Transactions

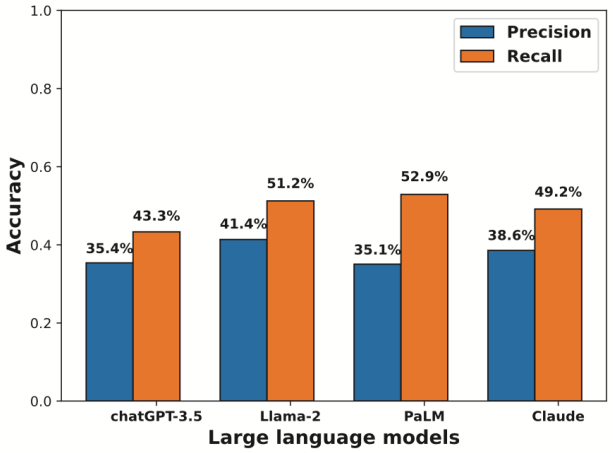
Attempts

a) **Streamlined Custom Configuration**

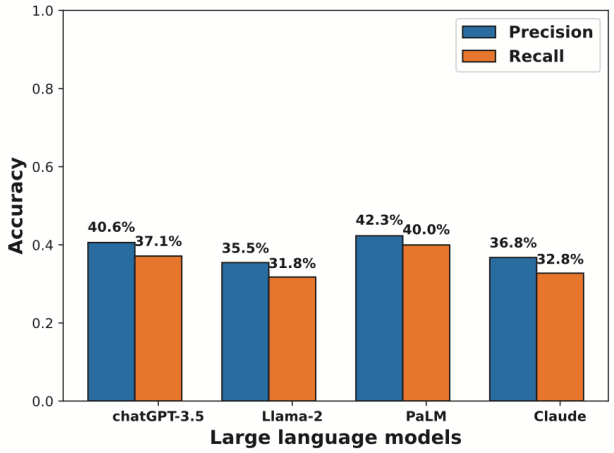
b) **Intended Transactions**

LLM Optimization

General-purpose LLMs *fail* in domain-specific tasks,
e.g., IoT fuzzing [S&P'24]



(a) cmd identification accuracy



(b) Format inference accuracy

General-purpose LLM does not work well for understanding IoT protocols [S&P'24]

Context matters!

Jincheng Wang, Le Yu, and Xiapu Luo. 2024. LLMIF: Augmented Large Language Model for Fuzzing IoT Devices. In Proc. of IEEE S&P.

LLM Optimization

"- **Provide reference text**

Language models can confidently invent fake answers, especially when asked about esoteric topics or for citations and URLs. In the same way that a sheet of notes can help a student do better on a test, ***providing reference text to these models*** can help in answering with fewer fabrications."

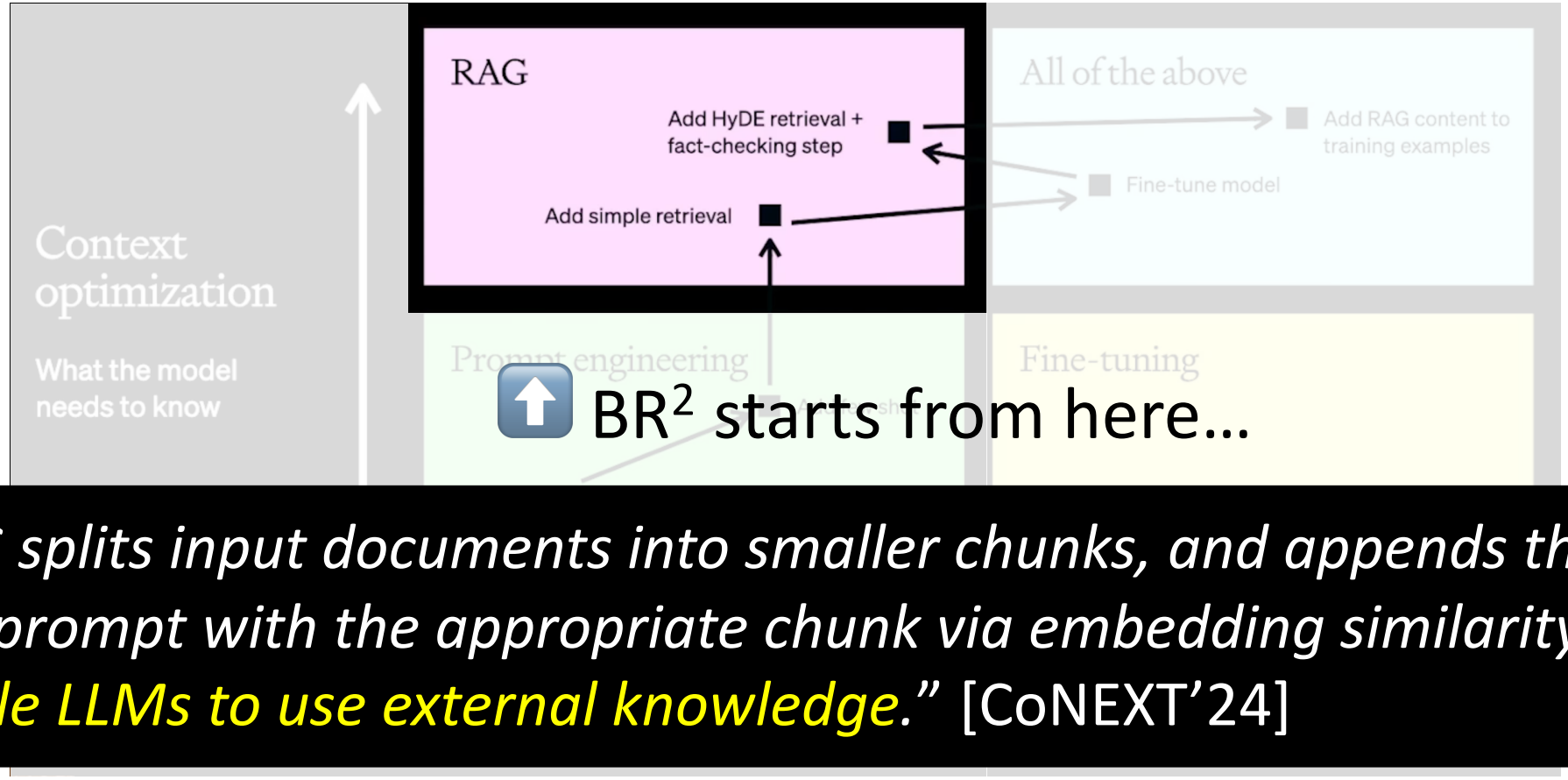
--*Six strategies for getting better results* [OpenAI]

Context matters!

https://platform.openai.com/docs/guides/prompt-engineering

LLM Optimization

The optimization flow of LLMs [OpenAI DevDay'23]



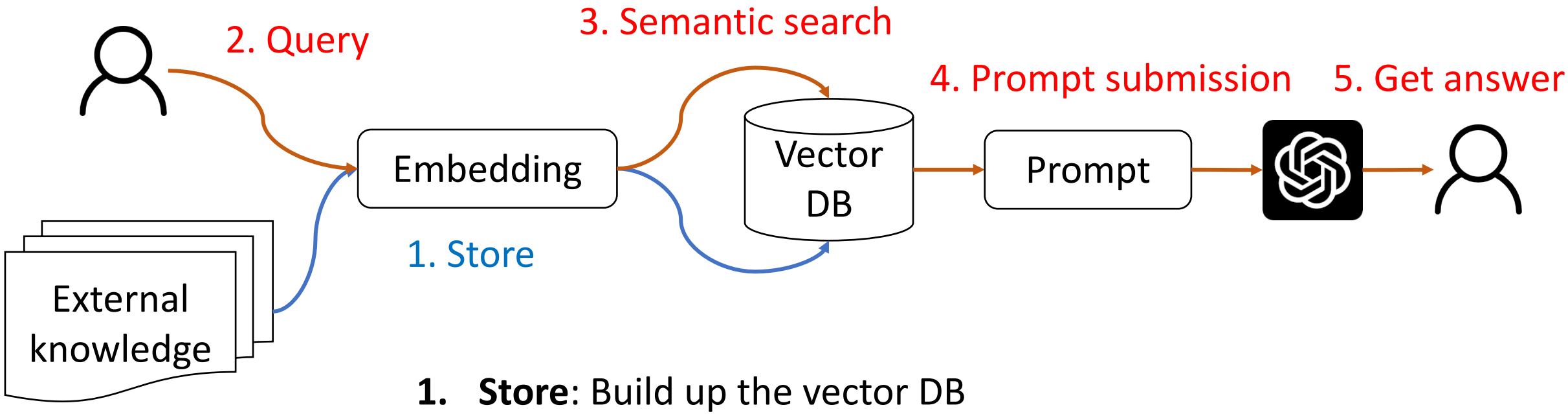
⬆️ BR^2 starts from here…

"RAG splits input documents into smaller chunks, and appends the user prompt with the appropriate chunk via embedding similarity to enable LLMs to use external knowledge." [CoNEXT'24]

Colin Jarvis and John Allard. 2023. Maximizing LLM Performance. https://www.youtube.com/watch?v=ahnGLM-RC1Y

Changjie Wang, Mariano Scazzariello, Alireza Farshin, et al. 2024. NetConfEval: Can LLMs Facilitate Network Configuration? In Proc. of CoNEXT.

Retrieval-Augmented Generation (RAG)



1. **Store**: Build up the vector DB
2. **Query**: Submit the request
3. **Semantic search**: Find the relevant contextual data
4. **Prompt submission**: Augment LLM prompts with contextual data
5. **Get answer**: Get the final answer

Retrieval-Augmented Generation (RAG)

A standard indexing pipeline:

1. Collect and load documents: e.g., by WebBaseLoader or DirectoryLoader
2. Split documents into chunks: e.g., recursively split by character, with chunk size 100 and overlap 20 by default
3. Encode the chunks into dense vector representations: with OpenAI embedding models, e.g., text-embedding-3-small (with dimensions of 512 or 1536)
4. Store the vectors: e.g., by Facebook AI Similarity Search (FAISS) with squared Euclidean (L2) distance
5. Vector store-backed retriever: find k nearest neighbors (KNN) to the query, k=4 by default

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proc. of NeurIPS.
Build a Retrieval Augmented Generation (RAG) App. https://python.langchain.com/v0.2/docs/tutorials/rag/
New embedding models and API updates. https://openai.com/index/new-embedding-models-and-api-updates/
Faiss. https://github.com/facebookresearch/faiss

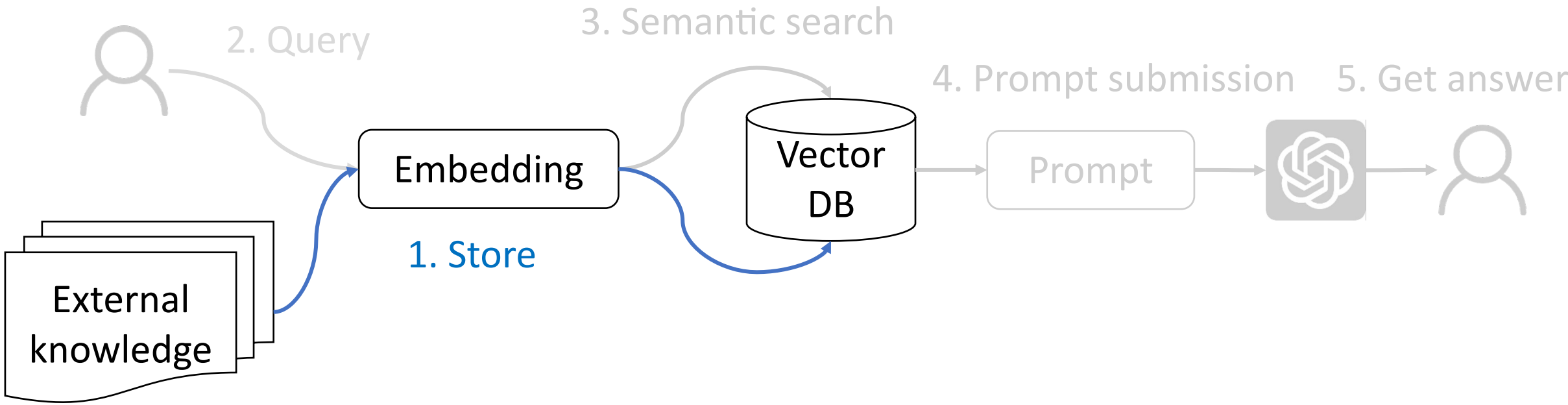
Retrieval-Augmented Generation (RAG)

SOTA optimizations to RAG pipeline: (our future work)

1. Enhanced data granularity (chunk size) [ICML'23]
2. Adding metadata (for better context)
3. Mixed retrieval (multimodal input, knowledge graph)
4. Enhanced self-reflection [NeurIPS'23, ICLR'24]
5. Chunk reranking (prioritize most relevant) [EMNLP Findings'23]

Freda Shi, Xinyun Chen, Kanishka Misra, et al. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. In Proc. of ICML.
Shu Liu, Asim Biswal, Audrey Cheng, et al. 2024. Optimizing llm queries in relational workloads. arXiv (2024)
Yunfan Gao, Yun Xiong, Xinyu Gao, et al. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv (2024).
Xin Cheng, Di Luo, Xiuying Chen, et al. 2023. Lift Yourself Up: Retrieval-augmented Text Generation with Self-Memory. In Proc. of NeurIPS.
Akari Asai, Zeqiu Wu, Yizhong Wang, et al. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In Proc. of ICLR.
Shengyao Zhuang, Bing Liu, Bevan Koopman, et al. 2023. Open-source Large Language Models are Strong Zero-shot Query Likelihood Models for Document Ranking. In Findings of EMNLP.

Retrieval-Augmented Generation (RAG)



Task: Jeopardy question generation

External knowledge: Wikipedia [NeurIPS'20]

Task: Multimodal generation

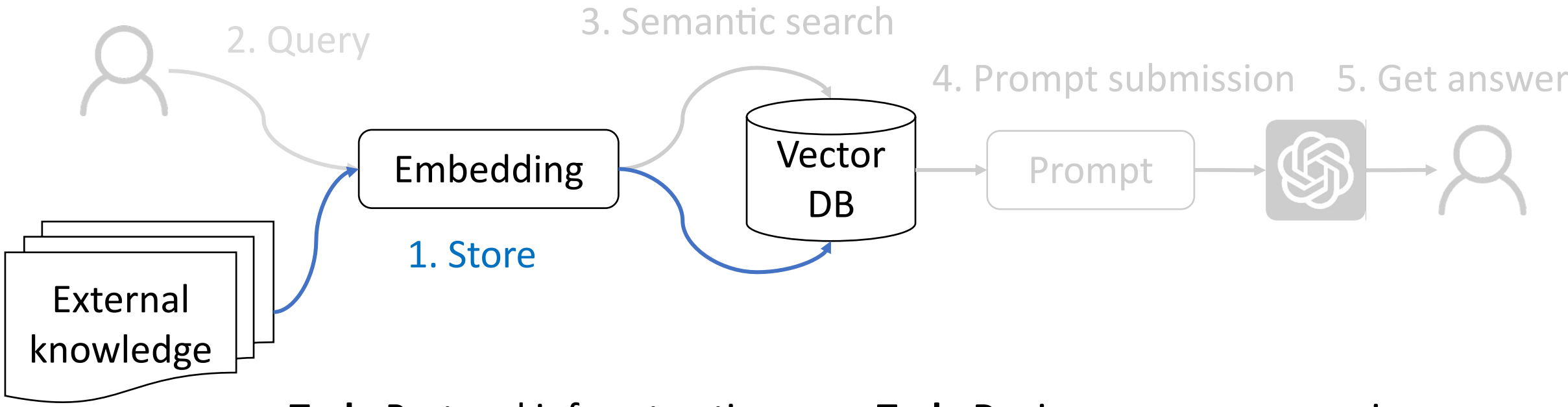
External knowledge: A combination of multimodal text and images [ICML'23]

Task: Program repair

External knowledge: Historic bug-fix code pairs [ESEC/FSE'23]

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proc. of NeurIPS.
Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, et al. 2023. Retrieval-Augmented Multimodal Language Modeling. In Proc. of ICML.
Matthew Jin, Syed Shahriar, Michele Tufano, et al. 2023. InferFix: End-to-End Program Repair with LLMs. In Proc. of ACM ESEC/FSE.

Retrieval-Augmented Generation (RAG)



Task: Protocol info extraction

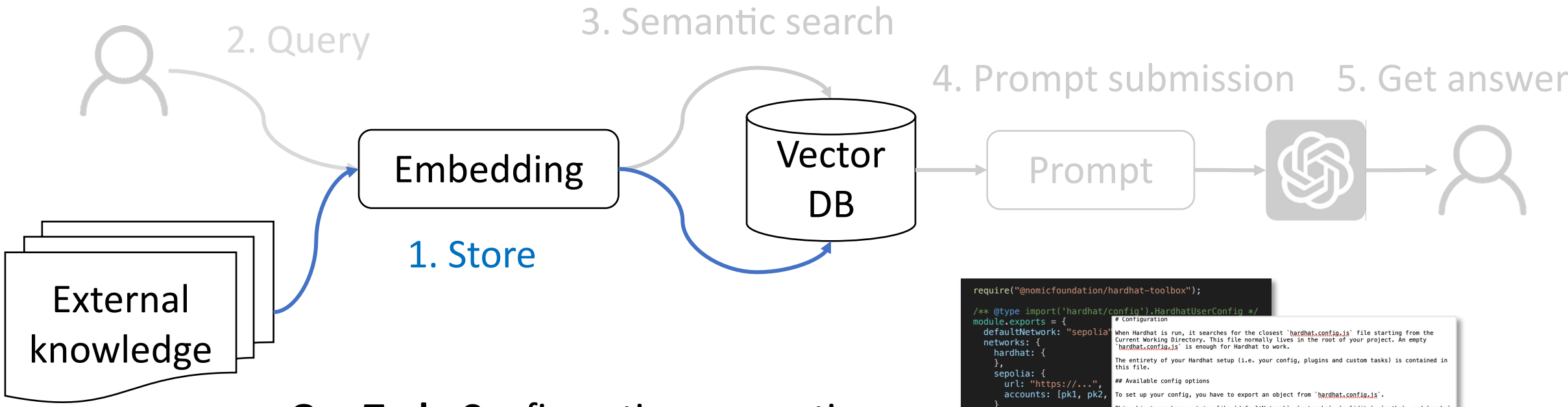
External knowledge: Specification document for IoT fuzzing [S&P'24]

Task: Device response reasoning

External knowledge: Crashing testing case and device response [S&P'24]

Jincheng Wang, Le Yu, and Xiapu Luo. 2024. LLMIF: Augmented Large Language Model for Fuzzing IoT Devices. In Proc. of IEEE S&P.

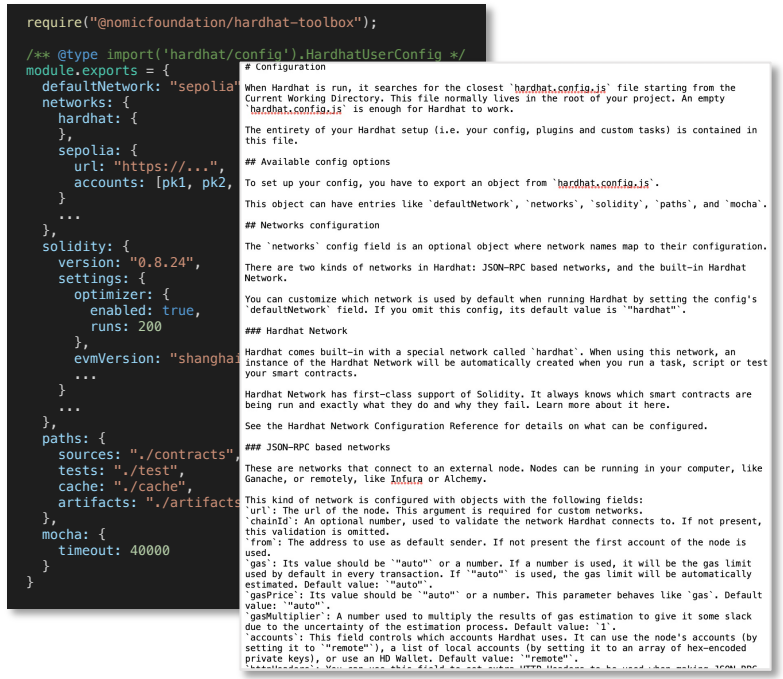
Retrieval-Augmented Generation (RAG)



Our Task: Configuration generation

External knowledge:

Hardhat configuration template and specification document



Attempt 1: LLM-assisted Configuration

A simple ***request*** in human language is all you need

Input Prompt:

You are a professional and experienced blockchain developer. Do not explain or calculate yourself, your answer should be in the same format as the provided context in JavaScript, and follow the requirements based on the question. No extra content except for the codes in your answer.

<context> {context} </context>

Question: {input}

Prompting instructions: background and requirements for the generated contents

Retrieval: retrieved external data that to be augmented into prompts, part of RAG.

Request: questions and demands from the developers

Attempt 1: LLM-assisted Configuration

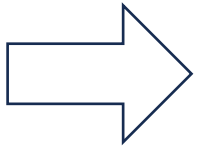
A simple ***request*** in human language is all you need

Input Prompt:

You are a professional and experienced blockchain developer. Do not explain or calculate yourself, your answer should be in the same format as the provided context in JavaScript, and follow the requirements based on the question. No extra content except for the codes in your answer.

<context> {context} </context>

Question: {"Compare the 4 EVM versions in throughput: Istanbul, London, Shanghai, and Cancun."}



```
require("@nomicfoundation/hardhat-toolbox");
/** @type
import('hardhat/config').HardhatUserConfig */
module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    hardhat: {
    },
  },
  solidity: {
    version: "0.6.12",
    settings: {
      evmVersion: "istanbul",
    }
  },
}
```

```
solidity: {
  version: "0.8.7",
  settings: {
    evmVersion: "london",
  }
},
```

```
  solidity: {
    version: "0.8.24",
    settings: {
      evmVersion: "shanghai",
    }
  },
```

```
solidity: {
  version: "0.8.25",
  settings: {
    evmVersion: "cancun",
  }
},
```

Attempt 1: LLM-assisted Configuration

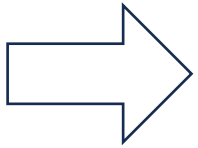
A simple ***request*** in human language is all you need

Input Prompt:

You are a professional and experienced blockchain developer. Do not explain or calculate yourself, your answer should be in the same format as the provided context in JavaScript, and follow the requirements based on the question. No extra content except for the codes in your answer.

<context> {context} </context>

Question: {"Try to configure Ethereum settings for a better throughput."}



```
require("@nomicfoundation/hardhat-toolbox");
/** @type
import('hardhat/config').HardhatUserConfig */
module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    hardhat: {
      gas: 8000000,
      gasPrice: 20000000000,
      gasMultiplier: 1,
      minGasPrice: 0,
      initialBaseFeePerGas: "100000",   default:
      blockGasLimit: 300_000_000,       30_000_000
      allowUnlimitedContractSize: true,
      mining: {                         default: false
        auto: false,
        interval: 1000,                 default: 3000
      }
    },
  },
  solidity: {
    version: "0.8.24",
    settings: {
      evmVersion: "shanghai",
    }
  },
}
```

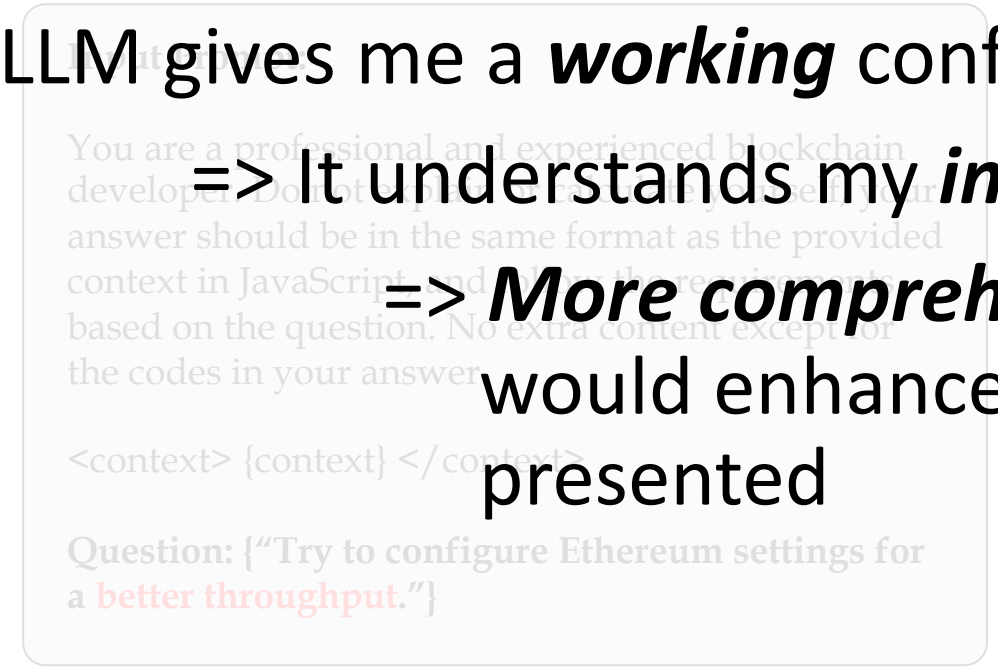
Attempt 1: LLM-assisted Configuration

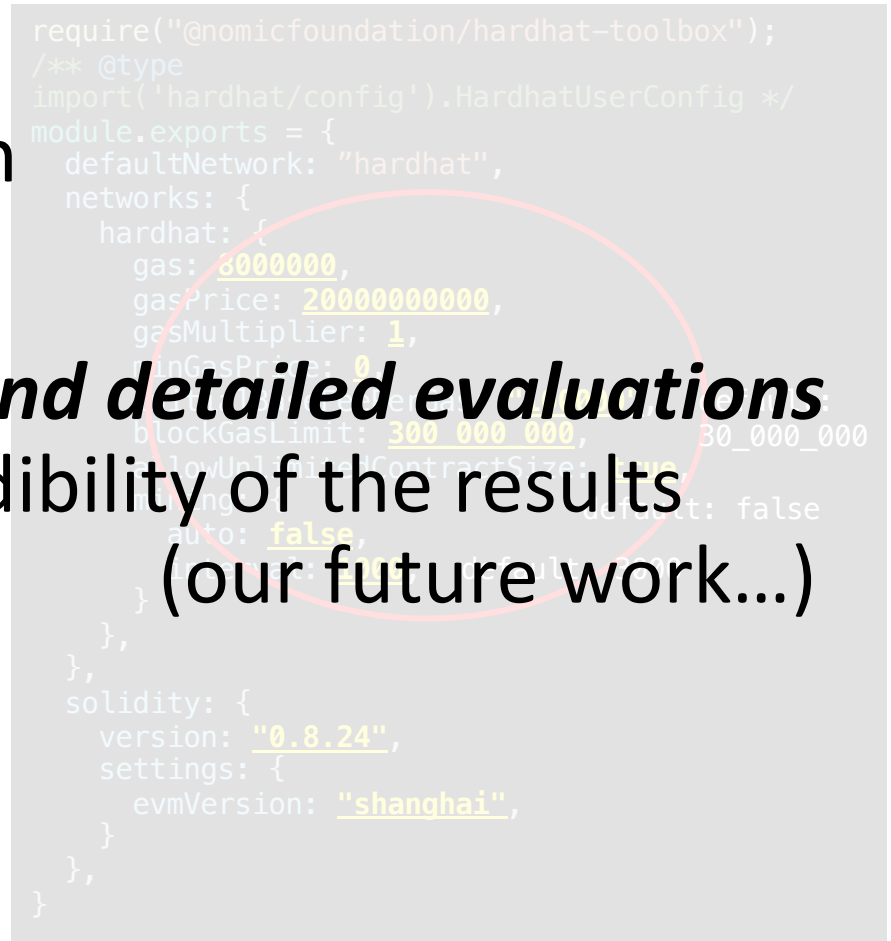
A simple ***request*** in human language is all you need

LLM gives me a ***working*** configuration

=> It understands my ***intention***

=> ***More comprehensive and detailed evaluations*** would enhance the credibility of the results presented (our future work…)





Attempts

Recall we have two attempts:

a) Streamlined Custom Configuration

b) Intended Transactions

Attempt 2: Intended Transactions

A series of prior works retrieve historical transactions to replay [ATC'20, ATC'21, S&P'22, TOSEM'22, ISSTA'23, TSC'24]

- real-world transactions, easy to access…

Chenxing Li, Peilun Li, Dong Zhou, et al. 2020. A Decentralized Blockchain with High Throughput and Fast Confirmation. In Proc. of USENIX ATC.
Yeonsoo Kim, Seongho Jeong, Kamil Jezek, et al. 2021. An Off-The-Chain Execution Environment for Scalable Testing and Profiling of Smart Contracts. In Proc. of USENIX ATC.
Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying Blockchain Extractable Value: How dark is the forest? In Proc. of IEEE S&P.
Siwei Wu, Lei Wu, Yajin Zhou, et al. 2022. Time-travel Investigation: Toward Building a Scalable Attack Detection Framework on Ethereum. TOSEM.
Huawei Huang, Xiaowen Peng, Jianzhou Zhan, et al. 2022. BrokerChain: A Cross-Shard Blockchain Protocol for Account/Balance-based State Sharding. In Proc. of INFOCOM.
Mingxi Ye, Yuhong Nan, Zibin Zheng, et al. 2023. Detecting State Inconsistency Bugs in DApps via On-Chain Transaction Replay and Fuzzing. In Proc. of ISSTA.
Yuan Huang, Rong Wang, Xiangping Chen, et al. 2024. Ethereum Transaction Replay Platform Based on State-wise Account Input Data. IEEE Transactions on Services Computing.

Attempt 2: Intended Transactions

More literature here,

- Banking transactions:
  - Generative adversarial network [UCAmI'22]
  - Arbitrary values [ICDE'22]
- Fixed transactions [DAPPS'22]
- Real-world workload traces [EuroSys'23]
- A workload generator to simulate concurrent users [TC'24]

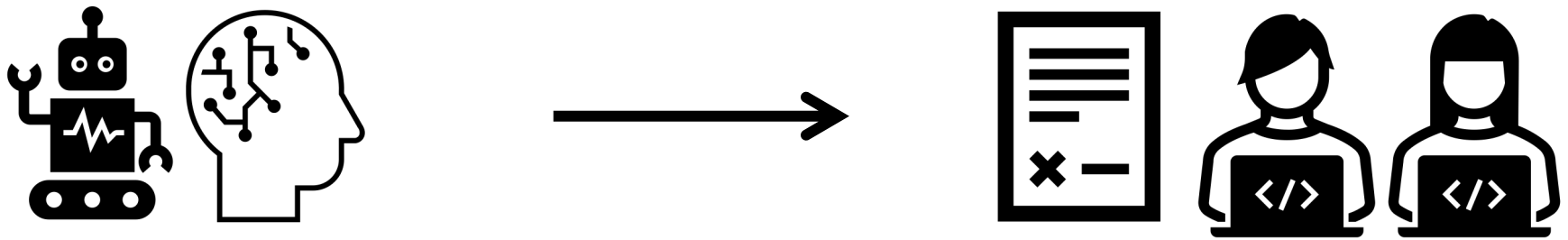
We want to build BR^2 to be able to ***respond to our intentions.***

Kyle Nickerson, Terrence Tricco, Antonina Kolokolova, et al. 2022. Banksformer: A Deep Generative Model for Synthetic Transaction Sequences. In Proc. of ECML PKDD.
Kamwoo Lee, et al. 2018. Generating Synthetic Bitcoin Transactions and Predicting Market Price Movement via Inverse Reinforcement Learning and Agent-Based Modeling. Journal of Artificial Societies and Social Simulation.
Pablo de Juan Fidalgo, Carmen Cámara, and Pedro Peris-Lopez. 2022. Generation and Classification of Illicit Bitcoin Transactions. In Proc. of UCAmI.
Wangze Ni, Peng Cheng, and Lei Chen. 2022. Mixing Transactions with Arbitrary Values on Blockchains. In Proc. of ICDE.
Bulat Nasrulin, Martijn De Vos, Georgy Ishmaev, et al. 2022. Gromit: Benchmarking the Performance and Scalability of Blockchain Systems. In Proc. of DAPPS.
Tien Tuan Anh Dinh, Ji Wang, Gang Chen, et al. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In Proc. of SIGMOD.
Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, et al. 2018. Untangling Blockchain: A Data Processing View of Blockchain Systems. IEEE Transactions on Knowledge and Data Engineering
Soujanya Ponnapalli, Aashaka Shah, Souvik Banerjee, et al. 2021. RainBlock: Faster Transaction Processing in Public Blockchains. In Proc. of USENIX ATC.
Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, et al. 2023. Diablo: A Benchmark Suite for Blockchains. In Proc. of EuroSys.
Mohammadreza Rasolroveicy, Wejdene Haouari, and Marios Fokaefs. 2024. BlockCompass: A benchmarking platform for blockchain performance. IEEE Transactions on Computers.

Attempt 2: Intended Transactions

Intention-based generation:

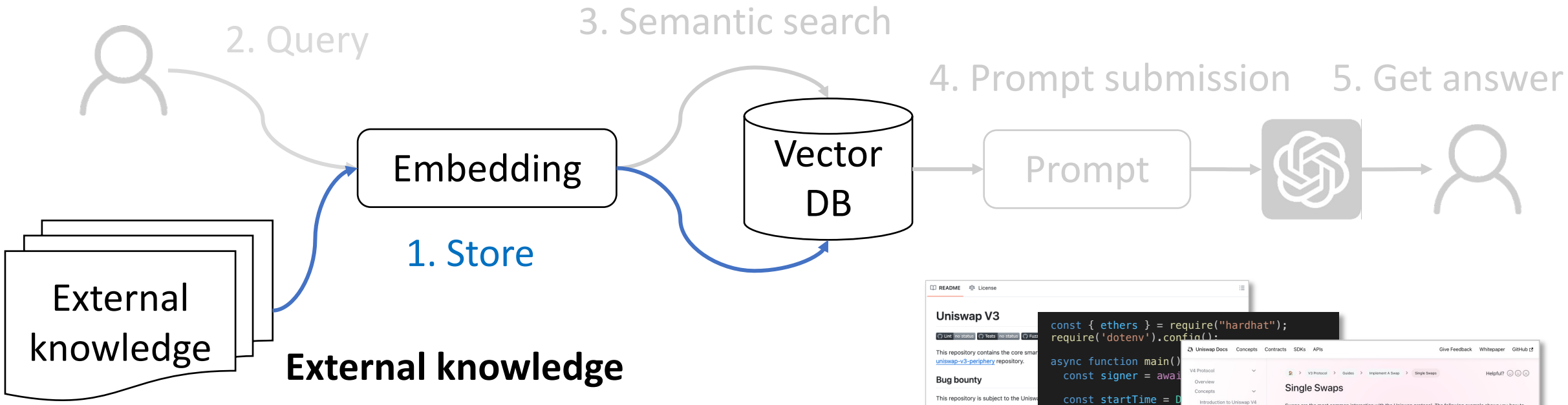
To trigger some specific functions of smart contracts, under certain circumstances, based on external context



LLMs

DApp smart contracts

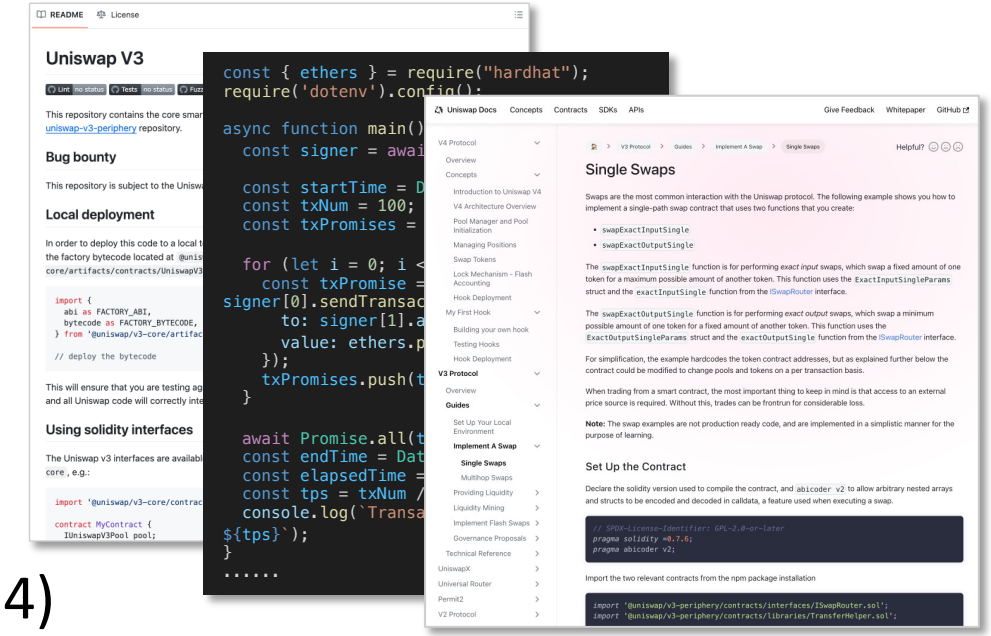
Attempt 2: LLM-assisted Transaction Synthesis



External knowledge

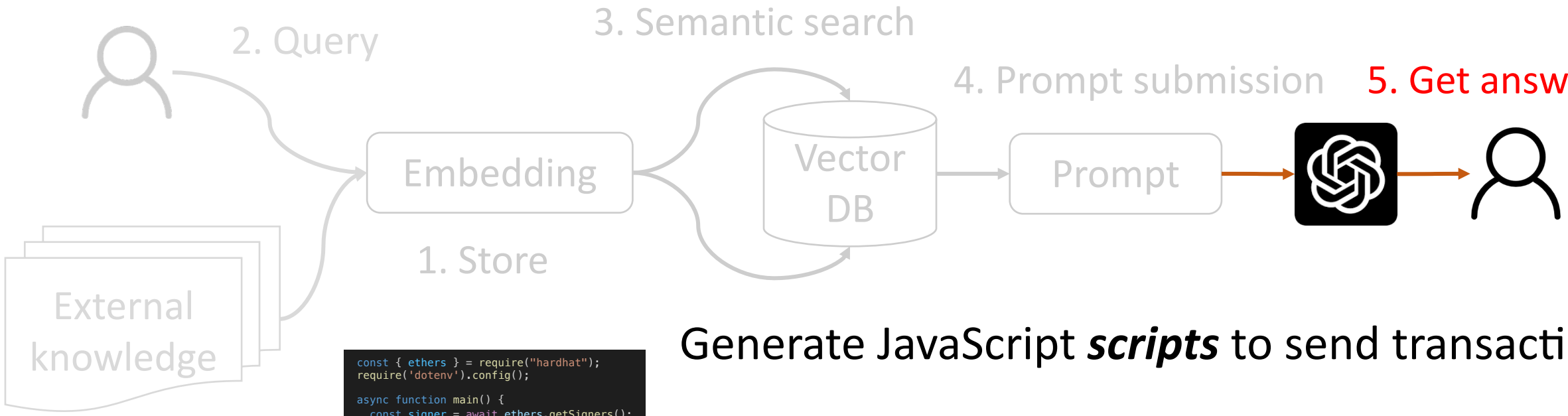
- DApp smart contract source code
- Transaction script templates for potential behaviors (e.g., swaps)
- Documentation (how to send tx)

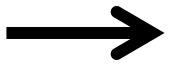
(Similar methodology following TSE'24)



Max Schäfer, Sarah Nadi, Aryaz Eghbali, et al. 2024. An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation. IEEE Transactions on Software Engineering.

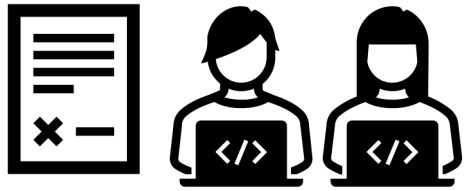
Attempt 2: LLM-assisted Transaction Synthesis





```
const { ethers } = require("hardhat");
require('dotenv').config();

async function main() {
  const signer = await ethers.getSigners();

  const startTime = Date.now();
  const txNum = 100;
  const txPromises = [];

  for (let i = 0; i < txNum; i++) {
    const txPromise =
signer[0].sendTransaction({
      to: signer[1].address,
      value: ethers.parseEther("0.1")
    });
    txPromises.push(txPromise);
  }

  await Promise.all(txPromises);
  const endTime = Date.now();
  const elapsedTime = endTime - startTime;
  const tps = txNum / (elapsedTime / 1000);
  console.log(`Transactions per second:
${tps}`);
}
.....
```

Generate JavaScript ***scripts*** to send transactions

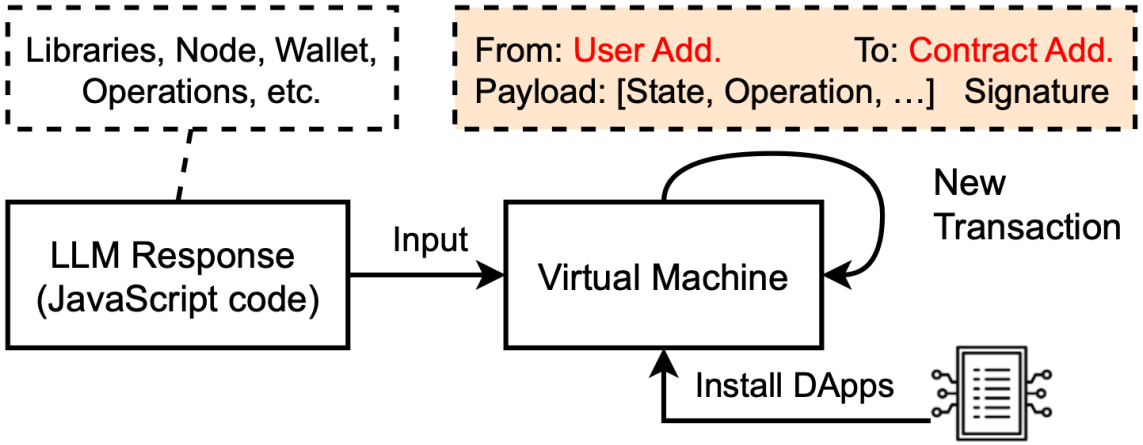


(web3.js & ethers.js libraries allow developers to interact with Ethereum easily)

Attempt 2: LLM-assisted Transaction Synthesis

Last Step: scripts are ***executed*** to send transactions

- just like people "send" transactions on real blockchains



Attempt 2: LLM-assisted Transaction Synthesis

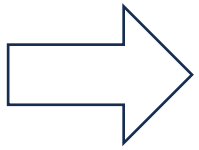
Recall that a simple ***request*** in human language is all you need

Input Prompt:

You are a professional and experienced blockchain developer. Do not explain or calculate yourself, your answer should be in the same format as the provided context in JavaScript, and follow the requirements based on the question. No extra content except for the codes in your answer.

<context> {context} </context>

Question: {"Try to configure Ethereum settings for a better throughput."}



```javascript
const { ethers } = require("hardhat");
require('dotenv').config();

async function main() {
  const signer = await ethers.getSigners();

  const startTime = Date.now();
  const txNum = 100000;
  const txPromises = [];

  for (let i = 0; i < txNum; i++) {
    const txPromise = signer[0].sendTransaction({
      to: signer[1].address,
      value: ethers.parseEther("0.1")
    });
    txPromises.push(txPromise);
  }

  await Promise.all(txPromises);
  const endTime = Date.now();
  const elapsedTime = endTime - startTime;
  const tps = txNum / (elapsedTime / 1000);
  console.log(`Transactions per second: ${tps}`);
}
```

BR² workflow:
1 prompt, 2 contexts, 2 generation outputs

Input Prompt:

You are a professional and experienced blockchain developer. Do not explain or calculate yourself, your answer should be in the same format as the provided context in JavaScript, and follow the requirements based on the question. No extra content except for the codes in your answer.

<context> {context} </context>

Question: {"Reproduce the Terra incident with oversell/bailout ratio of 0.5, for both UST and LUNA."}

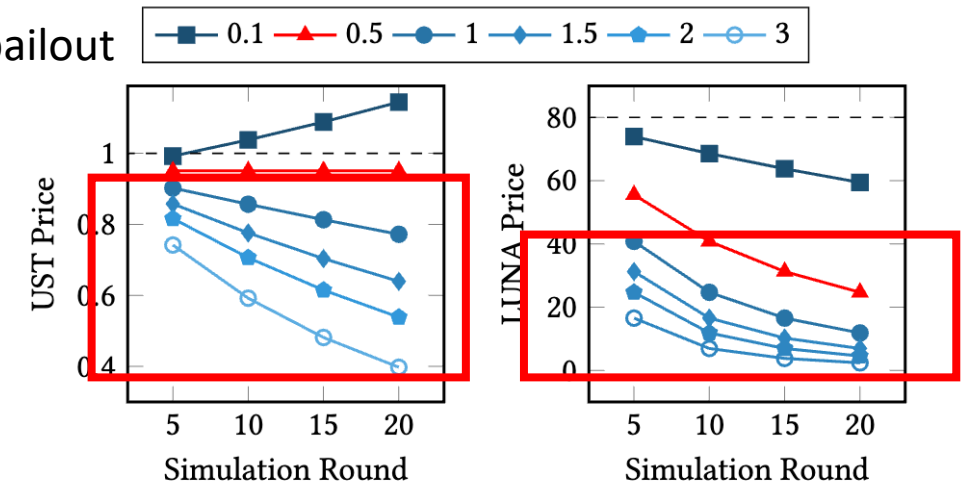
```
...
module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    hardhat: {
    },
  },
 solidity: {
    version: "0.8.24",
    settings: {
      evmVersion: "shanghai",
    }
  },
}
```

```
...
async function swapLuna() {

async function swapUst() {

async function swapUsdt() {
  ...
  const tx = await routerContract.exactInputSingle({
    tokenIn: process.env.USDT_ADDRESS,
    tokenOut: process.env.UST_ADDRESS,
    amountIn: ethers.utils.parseUnits("1000"), // swap 1000 USDT.
    ...
  });
}

const oversellLuna = 10; // for LUNA oversell
const oversellUst = 10;  // for UST oversell
const bailoutUst = 20;   // for UST bailout    default: 10

for (let i = 0; i < oversellLuna; i++) { swapLuna(); }
for (let j = 0; j < oversellUst; j++) { swapUst(); }
for (let k = 0; k < bailoutUst; k++) { swapUsdt(); }
```

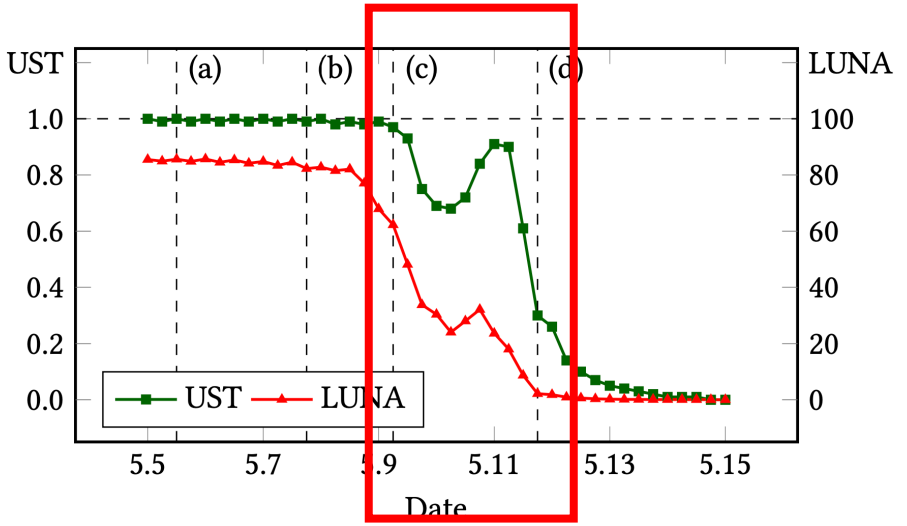
Demonstration: Terra UST/LUNA Incident

Demonstrate the depeg risk of Terra UST/LUNA (“*death spiral*”)

ratio of oversell/bailout



Simulation results



Real-world fluctuation

Demonstration: Beyond

- Vulnerability demonstration
  - demonstrate how and why the protocols can fail
- Ethereum Improvement Proposal (EIP) demonstration
  - demonstrate the potential consequences of “improvements”
- Tokenomics demonstration
  - demonstrate protocols from the economic aspect

Limitations and Future Plans

1. Build the benchmarking dataset
   - As done by other tasks: HumanEval, CausalBench, ReasonEval, NetConfEval [CoNEXT'24] …
2. Evaluate the quality of the output
   - Traditional measurements: accuracy score, F1 score… [S&P'24, NeurIPS'23, ICLR'23, ICSE'24]
   - AI-assisted evaluations
3. Optimize the standard RAG pipeline
   - Embedding model customization, fine-tuned retrieval distance metrics…
4. Extend similar work to Testnet
   - Realistic blockchain environment [FC'20, ATC'21]

Saad Ullah, Mingji Han, Saurabh Pujar, et al. 2024. LLMs Cannot Reliably Identify and Reason About Security Vulnerabilities (Yet?): A Comprehensive Evaluation, Framework, and Benchmarks. In Proc. of IEEE S&P.
Mark Chen, Jerry Tworek, Heewoo Jun, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv (2021).
Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, et al. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In Proc. of NeurIPS.
Ben Athiwaratkun, Sanjay Krishna Gouda, Zijian Wang, et al. 2023. Multi-lingual Evaluation of Code Generation Models. In Proc. of ICLR.
Xueying Du, Mingwei Liu, Kaixin Wang, et al. 2024. Evaluating Large Language Models in Class-Level Code Generation. In Proc. of ICSE.
Yunfan Gao, Yun Xiong, Xinyu Gao, et al. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv (2024).
Yu Zhou, Xingyu Wu, Beicheng Huang, et al. 2024. CausalBench: A Comprehensive Benchmark for Causal Learning Capability of Large Language Models. arXiv (2024).
Shijie Xia, Xuefeng Li, Yixin Liu, et al. 2024. Evaluating Mathematical Reasoning Beyond Accuracy. arXiv (2024).
Changjie Wang, Mariano Scazzariello, Alireza Farshin, et al. 2024. NetConfEval: Can LLMs Facilitate Network Configuration? In Proc. of CoNEXT.
Federico Franzoni, Ivan Abellan, and Vanesa Daza. 2020. Leveraging Bitcoin Testnet for Bidirectional Botnet Command and Control Systems. In Proc. of FC.
Yeonsoo Kim, Seongho Jeong, Kamil Jezek, et al. 2021. An Off-The-Chain Execution Environment for Scalable Testing and Profiling of Smart Contracts. In Proc. of USENIX ATC.

Acknowledgement

Shengchen LING

Research assistant (incoming PhD)

Department of Computer Science

City University of Hong Kong

Research interest: blockchain security & DeFi

Email: shengling2@cityu.edu.hk

Homepage: https://0xjackling.github.io/

Dr. Yufei CHEN

Postdoc

Department of Computer Science

City University of Hong Kong

Research interest: AI for security & AI safety

Email: yufeichen8@cityu.edu.hk

Homepage: https://yfchen1994.github.io

References

1. CoinDesk. Best Universities for Blockchain 2022. https://www.coindesk.com/layer2/2022/09/26/best-universities-for-blockchain-2022/
2. Coinbase. The rise of crypto in higher education. https://www.coinbase.com/blog/the-rise-of-crypto-in-higher-education
3. Blockchain and Cryptocurrency: What You Need to Know. https://drive.google.com/file/d/1sVo7QLkEvma__PCLLJib19aIeXRifkBW/view
4. Blockchain And Money. https://ocw.mit.edu/courses/15-s12-blockchain-and-money-fall-2018/
5. Decentralized Finance MOOC. https://defi-learning.org/f22
6. Wenliang Du, Honghao Zeng, Kyungrok Won. 2022. SEED Emulator: An Internet Emulator for Research and Education. In Proc. of HotNets.
7. J Scott Christianson. The Blockchain Game. https://www.instructables.com/The-Blockchain-Game/
8. Anders Brownworth. Blockchain Demo. https://andersbrownworth.com/blockchain/
9. Satoshi Nakamoto, 2008, Bitcoin: A Peer-to-Peer Electronic Cash System. Online at: https://bitcoin.org/bitcoin.pdf.
10. Liyi Zhou et al. 2023. SoK: Decentralized Finance (DeFi) Attacks. In Proc. of IEEE S&P.
11. Global Live Cryptocurrency Charts & Market Data. https://coinmarketcap.com/charts/
12. Ye Wang, Patrick Zuest, Yaxing Yao, et al. 2022. Impact and User Perception of Sandwich Attacks in the DeFi Ecosystem. In Proc. of ACM CHI.
13. List of Blockchain Attack Vectors and Vulnerabilities You Should Know, https://kingslanduniversity.com/blockchain-attack-vectors-vulnerabilities
14. What is a Reentrancy Attack in Smart Contracts and How to Prevent It? https://medium.com/blockchain-hacks/what-is-reentrancy-attack-in-smart-contracts-and-how-to-prevent-them-d65ad76dce5f
15. So You Want to Be a Security Expert? https://www.schneier.com/blog/archives/2012/07/how_to_become_a_1.html
16. Terra Docs. https://docs.terra.money/learn/protocol
17. Antonio Briola, David Vidal-Tomas, Yuanrong Wang, and Tomaso Aste. 2023. Anatomy of a Stablecoin's failure: the Terra-Luna case. Finance Research Letters.
18. DU09. 2022. Terra / Luna / UST Ecosystem Risks - How real are they? UST hodlers assemble!. https://www.reddit.com/r/terraluna/comments/s2bnbw/terra_luna_ust_ecosystem_risks_how_real_are_they/
19. SwissBorg. TerraLuna & UST - Risk Assessment. https://app.hubspot.com/documents/7219152/view/296921981?accessId=4d1141
20. Ethereum development environment for professionals, https://hardhat.org/
21. Documentation, https://hardhat.org/docs
22. Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In Proc. of USENIX OSDI.
23. Coinbase. What is mining? https://www.coinbase.com/en-sg/learn/crypto-basics/what-is-mining
24. Luca Beurer-Kellner, Martin Vechev, Laurent Vanbever, et al. 2022. Learning to Configure Computer Networks with Neural Algorithmic Reasoning. In Proc. of NeurIPS.
25. Sathiya Kumaran Mani, Yajie Zhou, Kevin Hsieh, et al. 2023. Enhancing Network Management Using Code Generated by Large Language Models. In Proc. of HotNets.
26. Changjie Wang, Mariano Scazzariello, Alireza Farshin, et al. 2024. NetConfEval: Can LLMs Facilitate Network Configuration? In Proc. of CoNEXT.
27. Prakhar Sharma and Vinod Yegneswaran. 2023. PROSPER: Extracting Protocol Specifications Using Large Language Models. In Proc. of HotNets.
28. Shengcheng Yu, Chunrong Fang, Yuchen Ling, et al. 2023. LLM for Test Script Generation and Migration: Challenges, Capabilities, and Opportunities. In Proc. of QRS.
29. Max Schäfer, Sarah Nadi, Aryaz Eghbali, et al. 2024. An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation. IEEE Transactions on Software Engineering.
30. Jincheng Wang, Le Yu, and Xiapu Luo. 2024. LLMIF: Augmented Large Language Model for Fuzzing IoT Devices. In Proc. of IEEE S&P.
31. Prompt engineering, https://platform.openai.com/docs/guides/prompt-engineering
32. Colin Jarvis and John Allard. 2023. Maximizing LLM Performance. https://www.youtube.com/watch?v=ahnGLM-RC1Y
33. Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proc. of NeurIPS.
34. Build a Retrieval Augmented Generation (RAG) App. https://python.langchain.com/v0.2/docs/tutorials/rag/
35. New embedding models and API updates. https://openai.com/index/new-embedding-models-and-api-updates/
36. Faiss. https://github.com/facebookresearch/faiss

References

37. Freda Shi, Xinyun Chen, Kanishka Misra, et al. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. In Proc. of ICML.
38. Shu Liu, Asim Biswal, Audrey Cheng, et al. 2024. Optimizing llm queries in relational workloads. arXiv (2024)
39. Yunfan Gao, Yun Xiong, Xinyu Gao, et al. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv (2024).
40. Xin Cheng, Di Luo, Xiuying Chen, et al. 2023. Lift Yourself Up: Retrieval-augmented Text Generation with Self-Memory. In Proc. of NeurIPS.
41. Akari Asai, Zeqiu Wu, Yizhong Wang, et al. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In Proc. of ICLR.
42. Shengyao Zhuang, Bing Liu, Bevan Koopman, et al. 2023. Open-source Large Language Models are Strong Zero-shot Query Likelihood Models for Document Ranking. In Findings of EMNLP.
43. Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, et al. 2023. Retrieval-Augmented Multimodal Language Modeling. In Proc. of ICML.
44. Matthew Jin, Syed Shahriar, Michele Tufano, et al. 2023. InferFix: End-to-End Program Repair with LLMs. In Proc. of ACM ESEC/FSE.
45. Chenxing Li, Peilun Li, Dong Zhou, et al. 2020. A Decentralized Blockchain with High Throughput and Fast Confirmation. In Proc. of USENIX ATC.
46. Yeonsoo Kim, Seongho Jeong, Kamil Jezek, et al. 2021. An Off-The-Chain Execution Environment for Scalable Testing and Profiling of Smart Contracts. In Proc. of USENIX ATC.
47. Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying Blockchain Extractable Value: How dark is the forest? In Proc. of IEEE S&P.
48. Siwei Wu, Lei Wu, Yajin Zhou, et al. 2022. Time-travel Investigation: Toward Building a Scalable Attack Detection Framework on Ethereum. TOSEM.
49. Huawei Huang, Xiaowen Peng, Jianzhou Zhan, et al. 2022. BrokerChain: A Cross-Shard Blockchain Protocol for Account/Balance-based State Sharding. In Proc. of INFOCOM.
50. Mingxi Ye, Yuhong Nan, Zibin Zheng, et al. 2023. Detecting State Inconsistency Bugs in DApps via On-Chain Transaction Replay and Fuzzing. In Proc. of ISSTA.
51. Yuan Huang, Rong Wang, Xiangping Chen, et al. 2024. Ethereum Transaction Replay Platform Based on State-wise Account Input Data. IEEE Transactions on Services Computing.
52. Kyle Nickerson, Terrence Tricco, Antonina Kolokolova, et al. 2022. Banksformer: A Deep Generative Model for Synthetic Transaction Sequences. In Proc. of ECML PKDD.
53. Kamwoo Lee, et al. 2018. Generating Synthetic Bitcoin Transactions and Predicting Market Price Movement via Inverse Reinforcement Learning and Agent-Based Modeling. Journal of Artificial Societies and Social Simulation.
54. Pablo de Juan Fidalgo, Carmen Cámara, and Pedro Peris-Lopez. 2022. Generation and Classification of Illicit Bitcoin Transactions. In Proc. of UCAmI.
55. Wangze Ni, Peng Cheng, and Lei Chen. 2022. Mixing Transactions with Arbitrary Values on Blockchains. In Proc. of ICDE.
56. Bulat Nasrulin, Martijn De Vos, Georgy Ishmaev, et al. 2022. Gromit: Benchmarking the Performance and Scalability of Blockchain Systems. In Proc. of DAPPS.
57. Tien Tuan Anh Dinh, Ji Wang, Gang Chen, et al. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In Proc. of SIGMOD.
58. Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, et al. 2018. Untangling Blockchain: A Data Processing View of Blockchain Systems. IEEE Transactions on Knowledge and Data Engineering
59. Soujanya Ponnapalli, Aashaka Shah, Souvik Banerjee, et al. 2021. RainBlock: Faster Transaction Processing in Public Blockchains. In Proc. of USENIX ATC.
60. Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, et al. 2023. Diablo: A Benchmark Suite for Blockchains. In Proc. of EuroSys.
61. Mohammadreza Rasolroveicy, Wejdene Haouari, and Marios Fokaefs. 2024. BlockCompass: A benchmarking platform for blockchain performance. IEEE Transactions on Computers.
62. Saad Ullah, Mingji Han, Saurabh Pujar, et al. 2024. LLMs Cannot Reliably Identify and Reason About Security Vulnerabilities (Yet?): A Comprehensive Evaluation, Framework, and Benchmarks. In Proc. of IEEE S&P.
63. Mark Chen, Jerry Tworek, Heewoo Jun, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv (2021).
64. Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, et al. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In Proc. of NeurIPS.
65. Ben Athiwaratkun, Sanjay Krishna Gouda, Zijian Wang, et al. 2023. Multi-lingual Evaluation of Code Generation Models. In Proc. of ICLR.
66. Xueying Du, Mingwei Liu, Kaixin Wang, et al. 2024. Evaluating Large Language Models in Class-Level Code Generation. In Proc. of ICSE.
67. Yu Zhou, Xingyu Wu, Beicheng Huang, et al. 2024. CausalBench: A Comprehensive Benchmark for Causal Learning Capability of Large Language Models. arXiv (2024).
68. Shijie Xia, Xuefeng Li, Yixin Liu, et al. 2024. Evaluating Mathematical Reasoning Beyond Accuracy. arXiv (2024).
69. Federico Franzoni, Ivan Abellan, and Vanesa Daza. 2020. Leveraging Bitcoin Testnet for Bidirectional Botnet Command and Control Systems. In Proc. of FC.

Thank you!

Questions?