

# BlockScan: Detecting Anomalies in Blockchain Transactions

Author: Jiahao Yu\*, Xian Wu\*, Hao Liu, Wenbo Guo, Xinyu Xing

Presenter: Jiahao Yu

Northwestern



 Meta

# The Problem: DeFi is Under Attack

Attacks against DeFi protocols (front-running, re-entrancy, etc.) cause billions in losses.  
We need to detect anomalous transactions *before* they do irreversible damage.

**Goal:** Identify transactions that exploit vulnerabilities, characterized by irregular method calls, abnormal parameters, and unusual operations.

# Why Existing Methods Fail

## Rule-Based & Traditional ML

**Rule-Based:** Cannot generalize to new, unseen attack patterns.

**Traditional ML (GMM, LSTM):** Fail to capture the complex, high-dimensional, and sequential nature of transaction data.

## Vanilla LLMs (e.g., GPT-4/5)

**Wrong Tool for the Job:** Off-the-shelf models don't understand the unique *data structure* of a transaction.

**Tokenization Failure:** They treat critical hashes (like `0x9fa0bc94`) and large numbers as generic text, losing all semantic meaning.

# Our Solution: BlockScan

## **A Customized Transformer for Transactions**

A BERT-style (RoBERTa) model, pretrained with Masked Language Modeling (MLM), to learn the "normal" patterns of benign transactions. Anomalies are transactions the model fails to reconstruct.

# Core Challenge: Data is Multi-Modal

**A blockchain transaction isn't just text. It's a complex, multi-modal structure.**

- **Hashes/Addresses:** e.g., `0x4deca5...`
- **Large Numbers:** e.g., `1962908` (gas)
- **Text Logs:** e.g., "Program... consumed..."

# Key Design 1: Multi-Modal Tokenizer

1. "type": "CALL",	<u>[START]</u>	<u>[CALL]</u>	<u>0xc1f351...5d0</u>	<u>0x4deca5...bac</u>	<u>0x9fa0bc94</u>	
2. "from": "0xc1f351...5d0",	start indicator of the calling	1. call indicator	2. from address	3. to address	4. function id	
5. "gas": 1962908,	<u>0x000000...39c</u>	<u>0x000000...000</u>	<u>[INs]</u>	<u>data</u>	<u>0x476f76...000</u>	
3. "to": "0x4deca5...bac",	5. gas 1962908 converted to hex	6. value 0 converted to hex	7. input indicator	7. input type and data		
4. "func": "0x9fa0bc94",	<u>address</u>	<u>0x000000...5d0</u>	<u>data</u>	<u>.....</u>	<u>[OUTs]</u>	<u>data</u>
7. "args": [...],						
"output": [{ "type": "data",						
8. "data": "0x000000...009"}],						
"calls": [{						
10. {						
"type": "DELEGATECALL",						
"from": "0x4deca5...bac",						
"gas": 1930278,	<u>0x000000...009</u>	<u>[logs]</u>	<u>"Program PhoeNi...units"</u>	<u>[END]</u>		
"to": "0x35dd16...5e8",	8. output type and data	9. log indicator	9. log messages	end indicator of the calling		
"func": "0x9fa0bc94",	<u>[START]</u>	<u>[DELEGATECALL]</u>	<u>[00V]</u>	<u>0x35dd16...5e8</u>	<u>0x9fa0bc94</u>	
"args": [...],	10. subsequent call's information	out of vocabulary				
"output": [...],	<u>0x000000...426</u>	<u>[NONE]</u>	<u>[INs]</u>	<u>data</u>	<u>0x476f76...000</u>	<u>.....</u>
"calls": [...],						
"logs": [...]						
"logMessages": [						
9. "Program PhoeNi... invoke [2]",						
"Program PhoeNi... consumed						
none compute units",						
6. "value": 0	<u>data</u>	<u>0x000000...009</u>	<u>[END]</u>	<u>[START]</u>	<u>[STATICCALL]</u>	<u>.....</u>

# Key Design 2: Model & Detection

## 1. Pre-training (MLM)

We train a RoBERTa-style model on millions of *benign* transactions.

## 2. Detection

Anomalies = High Reconstruction Error.

# Experimental Results

Method	k=10			k=15			k=20		
	FPR	Recall	Precision	FPR	Recall	Precision	FPR	Recall	Precision
<b>BlockGPT</b>	0.47%	16.67%	30%	0.73%	22.22%	26.67%	1%	27.78%	25%
<b>Doc2Vec</b>	0.67%	0%	0%	1%	0%	0%	1.13%	0%	0%
<b>GPT-4o</b>	0.67%	0%	0%	1%	0%	0%	1.13%	0%	0%
<b>Heuristic</b>	0.67%	0%	0%	1%	0%	0%	1.13%	0%	0%
BlockScan	<b>0.13%</b>	<b>44.44%</b>	<b>80%</b>	<b>0.2%</b>	<b>66.67%</b>	<b>80%</b>	<b>0.47%</b>	<b>72.22%</b>	<b>65%</b>

Table 1: Performance comparison with different  $k$  values for Solana.

Method	k=5			k=10			k=15		
	FPR	Recall	Precision	FPR	Recall	Precision	FPR	Recall	Precision
<b>BlockGPT</b>	0.14%	40%	80%	0.42%	70%	70%	0.99%	80%	53.33%
<b>Doc2Vec</b>	0.56%	10%	20%	1.12%	20%	20%	1.83%	20%	13.33%
<b>GPT-4o</b>	0.28%	30%	60%	0.98%	30%	30%	1.55%	40%	26.67%
<b>Heuristic</b>	0.14%	40%	80%	0.42%	70%	70%	1.13%	70%	46.67%
BlockScan	<b>0%</b>	<b>50%</b>	<b>100%</b>	<b>0.28%</b>	<b>80%</b>	<b>80%</b>	<b>0.97%</b>	<b>80%</b>	<b>53.33%</b>

Table 2: Performance comparison with different  $k$  values for Ethereum.



# Thank you!

Code: [https://github.com/nuwuxian/tx\\_fm](https://github.com/nuwuxian/tx_fm)