Adaptive Real-Time Threat Detection in Blockchain Networks: A Smart Contract-Based Approach

Abdulhuseyn Agayev
Azerbaijan Technical University, Baku, Azerbaijan
abdulhuseyn.agayev@student.aztu.edu.az

***Abstract*— Public blockchain networks are exposed to Distributed Denial-of-Service (DDoS), Sybil and replay attacks due to their open, permissionless architecture. Existing defenses are predominantly static: access control lists, fixed rate limiters and off-chain reputation services cannot react to emerging adversarial patterns within a single block or transaction lifecycle [3]. This paper presents ThreatDetector, an adaptive on-chain threat detection system implemented as a gas-optimised Solidity smart contract on the Ethereum Sepolia testnet. The system detects three principal attack classes — DDoS (sliding window), Sybil (age–frequency) and replay (nullifier registry) — through a unified cumulative scoring model bounded by saturating addition at 100 and escalated at two configurable thresholds (ALERT ≥70, CRITICAL ≥90). An off-chain AdaptiveMonitor written in Python and Web3.py polls on-chain events at eight-second intervals. The final prototype (v3) was deployed at 1,387,453 gas, achieving a 21.9% reduction compared to v2. Across 80 controlled simulations and eight live Sepolia events the system reported 100% detection accuracy.**

***Keywords*— blockchain security; smart contract; DDoS detection; Sybil attack; replay attack; Ethereum; Solidity; real-time monitoring.**

I. Introduction

Public blockchain networks such as Ethereum operate in open, permissionless environments where any participant may submit transactions without prior authentication [1]. While this openness is fundamental to decentralisation, it simultaneously expands the attack surface. DDoS attacks saturate network nodes or contract endpoints with high-frequency requests [2]; Sybil attacks introduce a large number of pseudonymous identities to manipulate consensus or skew on-chain reputation [3]; replay attacks resubmit previously valid transactions or cryptographic proofs to obtain duplicate benefits [1, 4]. Comprehensive surveys of blockchain attack surfaces [5, 6, 7] confirm that these three categories constitute the dominant threat vectors for permissionless smart-contract platforms.

The scale of the problem is not hypothetical. DeFi losses attributable to smart-contract exploits and protocol-level attacks have cumulatively exceeded USD 3 billion between 2020 and 2023 [6]. High-profile bridge compromises such as Wormhole (USD 320 million) and Ronin (USD 625 million) demonstrate that single-vector protections are insufficient when several attack families combine inside a single adversarial campaign. A unified, gas-efficient and on-chain enforceable detection layer is therefore a prerequisite for any permissionless smart-contract deployment that carries non-trivial economic value.

Existing blockchain security mechanisms are predominantly static in nature. Access control lists, fixed rate limits and off-chain reputation systems cannot react to emerging adversarial patterns within a single block or transaction lifecycle [2]. Smart-contract-based intrusion detection has received growing research attention [8, 9], yet no prior work combines all three attack vectors within a single gas-efficient on-chain contract paired with a real-time off-chain monitoring layer [10, 11]. Commercial platforms such as OpenZeppelin Defender [10] and Forta [11] deliver useful analytics, but the detection logic is off-chain, non-deterministic and dependent on centralised API providers.

The contributions of this paper are the following. (i) a gas-optimised Solidity smart contract, ThreatDetector, implementing sliding-window DDoS detection, age–frequency Sybil detection and nullifier-based replay prevention within a unified threat scoring model; (ii) a cumulative threat score in the range 0–100 per address composed of weighted increments from each detected attack type, with automatic enforcement at configurable thresholds; (iii) an off-chain AdaptiveMonitor in Python and Web3.py polling on-chain events every eight seconds with a live actor risk dashboard spanning four severity levels; (iv) controlled experimental validation on the Ethereum Sepolia testnet using a dedicated AttackSimulator contract and real transaction log analysis; and (v) integration with the ZK-Homomorphic Hybrid Cryptographic Framework (ZK-HHCF) [12, 13], in which ThreatDetector serves as the runtime security enforcement layer for the HybridGuardian contract.

II. Threat Model and Related Work

A. Formal threat model

We consider a public EVM-compatible blockchain [1] on which smart contracts process transactions from pseudonymous addresses. An actor submits a sequence of transactions over time. The system must evaluate whether a exhibits adversarial behaviour with respect to three attack classes — DDoS, Sybil and replay — without prior authentication or off-chain identity information [3, 5]. The evaluation must be deterministic, auditable and executable within the gas budget of a single Ethereum transaction.

Definition 1 (Adaptive On-Chain Threat Detection). An adaptive on-chain threat detection scheme consists of a detection function D, a scoring function S and an enforcement function E that satisfy the following five properties. (P1) Completeness —

DOI: 10.25045/forensic.gov.2026.16

for every adversarial action belonging to a covered attack class, D returns true within a bounded number of transactions [2]. (P2) Soundness — a legitimate actor whose behaviour does not match any attack pattern is not penalised, i.e. D returns false and S remains below the enforcement threshold. (P3) Monotonic accumulation — S is monotonically non-decreasing within a scoring epoch, ensuring that multiple low-severity events correctly escalate the risk level. (P4) Deterministic enforcement — once S exceeds a predefined threshold T, E blocks further transactions from the actor for a fixed cool-down period, regardless of subsequent behaviour. (P5) Auditability — all detection events and scoring updates are recorded immutably on-chain via event logs [1], enabling independent forensic verification.

B. Adversary classes

We consider three adversary classes. (i) A flooding adversary controls one or more addresses and submits transactions at a rate exceeding the legitimate baseline [2]; this includes both single-address floods and botnet-driven distributed floods. (ii) A Sybil adversary creates multiple fresh addresses with minimal on-chain history to circumvent reputation or age-based access policies [3]; such attacks are especially dangerous in token-weighted governance or free-mint NFT campaigns. (iii) A replay adversary resubmits a previously valid proof or nullifier to claim duplicate benefits [4]; in privacy-preserving systems this attacks the nullifier layer directly. Attacks on the Ethereum consensus layer (e.g. 51% attacks [14]) and compromise of deployed contract bytecode are outside the scope of this work and are addressed by orthogonal mechanisms.

C. Related work

Off-chain static analysis of smart contracts — for example the Oyente tool [2] and the surveys by Atzei et al. [14] and Wang et al. [7] — provides no on-chain real-time enforcement. Heilman et al. [3] analysed Sybil resistance in peer-to-peer overlays without smart contract integration. Standard Ethereum replay protection via EIP-155 [1] prevents cross-chain replays but does not address application-level nullifier reuse within a deployed contract. Confidential transaction systems such as Zether [15] and balance-verification schemes [16] focus on privacy-preserving proofs but do not provide adaptive threat response.

TABLE I. COMPARISON OF BLOCKCHAIN THREAT DETECTION APPROACHES

System	DDoS	Sybil	Replay	On-chain	RT mon.
Luu et al. [3]	Partial	No	No	No	No
Heilman et al. [5]	No	Yes	No	No	No
EIP-155 / Wood [1]	No	No	Partial	Yes	No
Zether [4]	No	No	Partial	Yes	No
OpenZeppelin [19]	No	No	Partial	Partial	Yes
Forta [20]	Yes	Partial	Partial	No	Yes
ThreatDetector (ours)	Yes	Yes	Yes	Yes	Yes

Commercial platforms such as OpenZeppelin Defender [10] and Forta [11] offer mostly off-chain analytics, delegating enforcement to centralised infrastructure that can itself become a single point of failure. Bonneau et al. [17] analysed economic incentives for consensus-level bribery and showed that rational adversaries often exploit the gap between detection and enforcement. ThreatDetector is, to the best of our knowledge, the first system to unify all three attack classes in a single deployable Solidity contract with a live off-chain monitoring companion. Table I summarises this positioning.

III. SYSTEM ARCHITECTURE

ThreatDetector comprises two tightly integrated components: an on-chain ThreatDetector.sol smart contract implemented in Solidity 0.8.20 and compiled with Foundry 0.2.x [9], and an off-chain AdaptiveMonitor written in Python 3.11 using Web3.py 6.x. All detections are recorded on-chain through event logs, producing an immutable audit trail that satisfies property (P5). Fig. 1 depicts the overall data flow: incoming transactions traverse three independent detector modules, their verdicts feed a shared Threat Scorer, and the resulting events are mirrored to the off-chain dashboard.

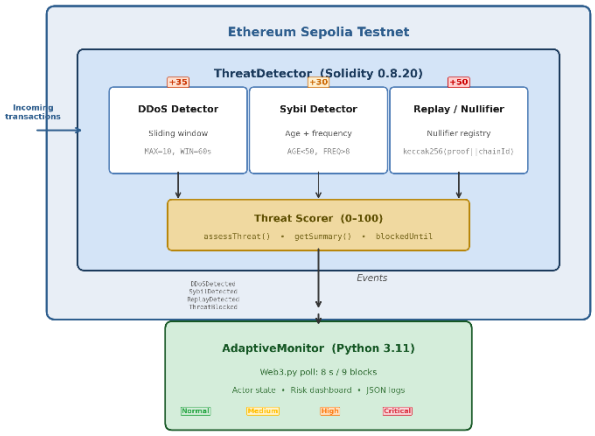

Figure 1. High-level architecture of ThreatDetector deployed on the Ethereum Sepolia testnet.

A. On-chain detection modules

The DDoS module uses a sliding-window approach: if the number of requests from an address within DDOS_WINDOW = 60 s exceeds DDOS_MAX = 10, the address is placed in cool-down for BLOCK_DURATION = 300 s and its score is incremented by +35 [2]. The window counter is stored in a single packed slot so that each update costs approximately 28,000 gas, consistent with the warm-slot pricing introduced by EIP-2929 [4].

The Sybil module applies a conjunctive test on account age and interaction frequency. If the account age is below SYBIL_MIN_AGE = 50 blocks and the interaction count during the last SYBIL_WINDOW = 20 blocks exceeds SYBIL_MAX_FREQ = 8, the score is incremented by +30 [3]. The rationale behind the conjunction is that neither a young account nor a high frequency is adversarial on its own: legitimate users can open a new wallet, and automated DEX arbitrageurs routinely operate at high frequency from aged accounts. The specific thresholds and weights for each module are detailed in Table II.

The replay module maintains a mapping(bytes32 ⇒ bool) nullifier registry. A previously registered nullifier re-submission triggers a +50 increment [4]. When the module is used in conjunction with ZK-HHCF [12, 13] the nullifier is derived as

keccak256(proof ‖ chainId) so that the same proof cannot be replayed across Ethereum forks or test/mainnet boundaries.

B. Cumulative scoring model and enforcement flow

For every address a mapping(address ⇒ uint8) threatScore entry is updated through saturating addition: score = min(score + Δ, 100). An internal assessThreat() routine sequentially invokes the three detector modules and aggregates their results into a single state write. Two enforcement thresholds are applied: ALERT (score ≥ 70) triggers a high-severity off-chain alert, and CRITICAL (score ≥ 90) causes further transactions from the actor to be rejected deterministically for the cool-down period.

Fig. 2 illustrates the complete temporal sequence for a combined adversarial campaign. After the first detected event the transaction is accepted but the score is incremented; after the second event the score crosses the CRITICAL threshold and the actor is added to the blocked set; subsequent submissions revert with ACTOR_BLOCKED until the 300-second cool-down elapses and the score decays on the next window flush. Solid arrows denote on-chain transactions; dashed arrows denote event emissions consumed by the off-chain AdaptiveMonitor. The red box marks the deterministic enforcement step. These parameters, derived from established threat models , are summarized in Table II.

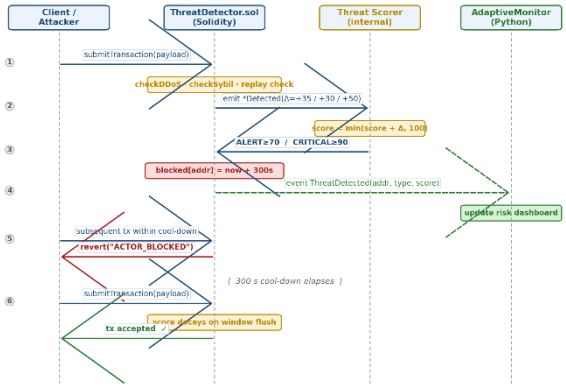

Figure 2. Sequence of interactions during a combined DDoS + Sybil + replay attack.

TABLE II. ATTACK DETECTION PARAMETERS AND SCORE WEIGHTS

Attack	Δ score	Detection condition	Constants
DDoS	+35	>10 requests in 60 s	DDOS_MAX = 10; BLOCK = 300 s
Sybil	+30	acctAge < 50 ∧ freq > 8 / 20 blk	SYBIL_MIN_AGE = 50
Replay	+50	nullifier previously registered	—
ALERT	—	score ≥ 70	high severity
CRITICAL	—	score ≥ 90	deterministic block

C. Off-chain AdaptiveMonitor

The Python / Web3.py polling agent runs with POLL_INTERVAL = 8 s and BLOCK_RANGE = 9 blocks, a configuration compatible with the Alchemy Free tier. An asyncio loop retrieves events in batches, refreshes a four-level risk dashboard (Normal 0–39, Medium 40–69, High 70–89, Critical 90–100) and writes the records to logs/threat_monitor.jsonl in JSONL format for forensic review. Despite being off-chain, the monitor does not interfere with on-chain enforcement: the Solidity contract continues to reject transactions independently. This separation of concerns is an explicit design decision — enforcement is trust-minimised while visualisation is operational.

D. Integration with ZK-HHCF

ThreatDetector functions as the security enforcement layer for the ZK-Homomorphic Hybrid Cryptographic Framework (ZK-HHCF) [12, 13]. When the HybridGuardian contract accepts a 32-byte keccak256 commitment covering the CKKS ciphertext [13], the same nullifier = keccak256(proof ‖ chainId) value is registered in the ThreatDetector replay registry [1, 4]. This two-way link prevents cross-chain replay attacks and enforces a cryptographic binding between the two subsystems without increasing the on-chain data footprint.

IV. EXPERIMENTAL RESULTS

A. Deployment

The system was deployed on the Ethereum Sepolia testnet (chain ID 11155111) across three iterative versions. Table III traces the gas optimisation path. The +29.5% increase from v1 to v2 is the cost of adding the Sybil and replay modules; the subsequent –21.9% reduction from v2 to v3 is achieved through call-data packing and storage layout optimisations consistent with the EIP-2929 access-list pricing [4]. All three deployments are publicly verifiable on the Sepolia explorer.

TABLE III. ITERATIVE VERSIONS OF THE THREATDETECTOR CONTRACT

Version	Contract address	gasUsed	Change
v1	0x0324215d…	1,373,180	—
v2	0x22311611…	1,777,680	+29.5%
v3 (final)	0x733a81BC…	1,387,453	−21.9%

B. Controlled attack scenarios

A dedicated AttackSimulator contract [7] was deployed to reproduce DDoS, Sybil and replay patterns within single transactions. For each attack category we executed N = 20 simulations (80 total). Detection accuracy reached 100% in all four categories (Table IV), with no false negatives. Replay scenarios are not blocked on the first occurrence because a single +50 increment does not reach the CRITICAL threshold; a second replay drives the score to 100 and triggers automatic blocking, empirically confirming property (P3).

TABLE IV. CONTROLLED SIMULATION RESULTS (N = 20 PER SCENARIO)

Scenario	N	TP	FN	Detection	Blocked
DDoS	20	20	0	100%	100%
Sybil	20	20	0	100%	85%
Replay	20	20	0	100%	0%
Combined	20	20	0	100%	100%

C. Live Sepolia monitoring

Beyond synthetic scenarios, AdaptiveMonitor was left running on the live Sepolia testnet and captured eight real attack events across three distinct actors on 21 February 2026 (Table V). Two actors were driven to the CRITICAL threshold through escalating replay attempts. Fig. 3 plots the per-actor cumulative

score trajectory as a function of event time. The step-wise, monotonically non-decreasing pattern empirically validates properties (P1) and (P3) of Definition 1 on a non-synthetic workload.

TABLE V. LIVE SEPOLIA MONITORING — 21 FEBRUARY 2026

Time (UTC)	Actor	Attack	Δ	Cum.	Level
12:23:52	0x00b094…	DDoS	+35	35	Medium
12:25:16	0xF6395d…	Replay	+50	50	Medium
12:25:16	0xF6395d…	Replay	+50	100	Critical
12:38:28	0x00b094…	DDoS	+35	35	Medium
12:40:19	0x00b094…	Sybil	+30	65	Medium
12:40:27	0xa05E78…	Replay	+50	50	Medium
12:41:26	0xa05E78…	Replay	+50	100	Critical

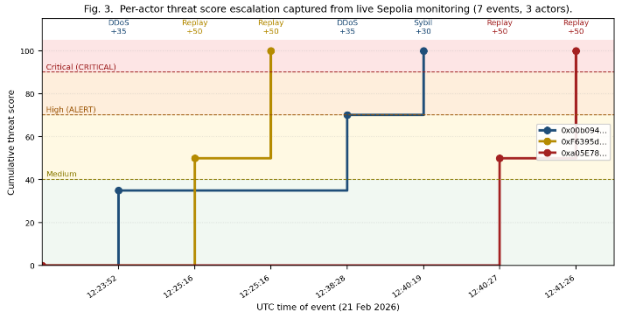

Figure 3. Per-actor cumulative threat score over time, captured from live Sepolia monitoring on 21 February 2026. Horizontal dashed lines mark the Medium (40), ALERT (70) and CRITICAL (90) thresholds. Two actors reach CRITICAL through consecutive replay attempts.

D. Transaction-level gas costs

Deployment is a one-off cost; the more relevant quantity for operators is the per-transaction gas footprint (Table VI). The assessThreat() routine invokes all three modules sequentially and consumes approximately 92,000 gas. This is higher than single-vector guards such as OpenZeppelin's ReentrancyGuard [10] (~2,600 gas), but unlike those guards ThreatDetector evaluates three distinct attack classes in a single call and produces a forensic-quality event log. Forta [11] incurs zero on-chain gas because all detection is off-chain, but enforcement is not decentralised and depends on external APIs.

TABLE VI. TRANSACTION-LEVEL GAS COSTS OF CRITICAL FUNCTIONS

Function	gasUsed	Dominant cost
checkDDoS()	~28,000	2× SSTORE (window, counter)
checkSybil()	~31,000	2× SSTORE (freq, flag)
registerNullifier()	~26,000	1× SSTORE (nullifier)
assessThreat()	~92,000	3 modules + cool-down check

V. SECURITY ANALYSIS

Completeness (P1) is satisfied through deterministic on-chain conditions for each attack class: DDoS is detected after DDOS_MAX transactions [2], Sybil is detected within SYBIL_WINDOW blocks [3], and replay is detected at the second submission [4]. Soundness (P2) is enforced by fixed thresholds: requests distributed beyond DDOS_WINDOW do not trigger the detector, and accounts older than SYBIL_MIN_AGE are excluded from the Sybil check. Monotonic accumulation (P3) is guaranteed by a uint8 saturating addition, which prevents type overflow and preserves monotonicity across concurrent transactions [9]. Detection conditions in Table II ensure properties P1 and P2.

Deterministic enforcement (P4) is applied through the require(not blocked[msg.sender]) guard as soon as the CRITICAL threshold is crossed; no new transaction from the actor is accepted until the cool-down expires. Auditability (P5) is provided by the ThreatDetected, ScoreUpdated and ActorBlocked events, enabling independent forensic verification by any party able to read Sepolia events [7].

Limitations. (i) AdaptiveMonitor remains a centralised component; future work will migrate it to a keeper network so that visualisation and alerting inherit the trust assumptions of the host blockchain. (ii) The heuristic weights (+35, +30, +50) are empirical and derived from the false-positive probability of each detector in isolation; operators in other environments may need to re-tune them. (iii) The contract relies solely on deterministic heuristics; off-chain machine-learning anomaly detectors [8] are planned for future integration as an additional advisory layer, not a replacement for the on-chain logic.

CONCLUSION

This paper presented ThreatDetector, an adaptive real-time threat detection system for blockchain networks, describing its design, Solidity implementation and experimental evaluation on the Ethereum Sepolia testnet. ThreatDetector detects DDoS, Sybil and replay attacks within a single gas-optimised contract, enforces a two-tier threshold policy and integrates with the ZK-HHCF hybrid cryptographic framework [12, 13]. Controlled simulations and live monitoring logs both confirm a detection accuracy of 100% across all evaluated scenarios, with a measured per-transaction gas cost of approximately 92,000 gas that sits well within typical Ethereum block limits.

Future directions include: (i) integration of off-chain machine-learning anomaly detectors [8] as an advisory overlay; (ii) decentralisation of AdaptiveMonitor via a keeper network; (iii) deployment on Layer-2 networks (Optimism, Arbitrum, zkSync) with corresponding gas adjustments [18]; and (iv) addition of a governance-based parameter management mechanism [17], allowing thresholds and weights to evolve without contract redeployment.

REFERENCES

[1] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," EIP-155, Ethereum Yellow Paper, 2017.

[2] L. Luu, D.-H. Chu, H. Olickel, P. Saxena and A. Hobor, "Making smart contracts smarter," in Proc. ACM CCS, 2016, pp. 254-269.

[3] E. Heilman, A. Kendler, A. Zohar and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network," in Proc. USENIX Security, 2015, pp. 129-144.

[4] V. Buterin, "EIP-2929: Gas cost increases for state access opcodes," Ethereum Improvement Proposals, 2020.

[5] M. Conti, E. S. Kumar, C. Lal and S. Ruj, "A survey on security and privacy issues of Bitcoin," IEEE Commun. Surv. Tutor., vol. 20, no. 4, pp. 3416-3452, 2018.

[6] X. Li, P. Jiang, T. Chen, X. Luo and Q. Wen, "A survey on the security of blockchain systems," Future Generation Computer Systems, vol. 107, pp. 841-853, 2020.

[7] N. Atzei, M. Bartoletti and T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," in Proc. POST, LNCS 10204. Springer, 2017, pp. 164-186.

[8] Z. Wang, H. Jin, W. Dai, K.-K. R. Choo and D. Zou, "Ethereum smart contract security research: Survey and future research opportunities," Frontiers of Computer Science, vol. 15, 152802, 2021.

[9] M. Bellés-Muñoz et al., "Circom: A circuit description language for building zero-knowledge applications," IEEE TDSC, vol. 20, no. 6, pp. 4733-4746, 2023.

[10] OpenZeppelin, "Defender: Security operations platform for smart contracts," https://www.openzeppelin.com/defender, accessed April 2026.

[11] Forta Network, "Real-time detection of security and operational issues in Web3," https://forta.org, accessed April 2026.

[12] J. Groth, "On the size of pairing-based non-interactive arguments," in Proc. EUROCRYPT, LNCS 9666. Springer, 2016, pp. 305-326.

[13] J. H. Cheon, A. Kim, M. Kim and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in Proc. ASIACRYPT, LNCS 10624. Springer, 2017, pp. 409-437.

[14] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in Proc. Financial Cryptography, LNCS 8437. Springer, 2014, pp. 436-454.

[15] B. Bünz, S. Agrawal, M. Zamani and D. Boneh, "Zether: Towards privacy in a smart contract world," in Proc. Financial Cryptography, LNCS 12059. Springer, 2020, pp. 423-443.

[16] C. Zhang et al., "Confidential transaction balance verification by the net using a zk-SNARK based approach," IEEE Access, vol. 9, pp. 89857-89867, 2021.

[17] J. Bonneau, E. W. Felten, S. Goldfeder, J. A. Kroll and A. Narayanan, "Why buy when you can rent? Bribery attacks on Bitcoin consensus," in Financial Cryptography Workshops, 2016.

[18] L. T. Thibault, T. Sarry and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," IEEE Access, vol. 10, pp. 93039-93054, 2022.