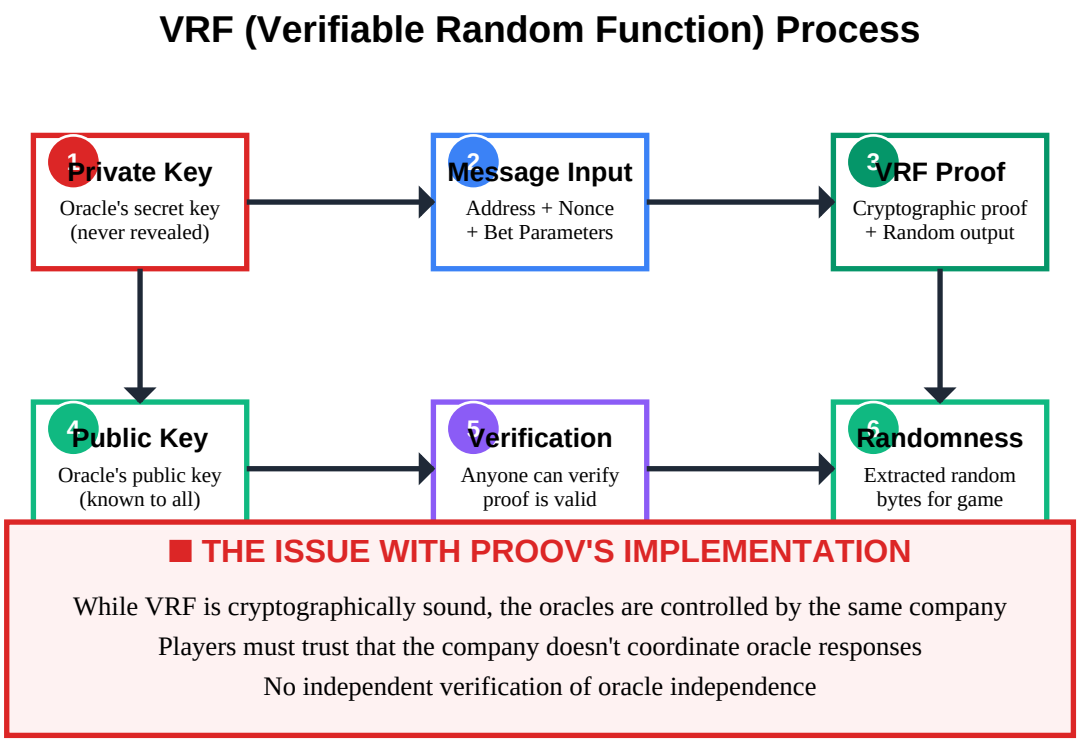
PROOV GAMBLING ALGORITHMS

Technical Analysis of Game Verification Systems

Executive Summary

This report analyzes the cryptographic algorithms and verification processes used by Proov Network for online gambling. While the mathematics are sound, the centralized oracle control presents significant trust assumptions for players.

VRF (Verifiable Random Function) Analysis



Technical Implementation:

```
def verify_vrf(pk: bytes, msg: bytes, proof: bytes): gamma = proof[:32] # VRF output point c16 = proof[32:48] # Challenge hash s = proof[48:] # Scalar response # Verify the proof using Ed25519 operations h = _hash_to_curve_tai(pk, msg) U = s·B − c·Y # Verification equation 1 V = s·H − c·Γ # Verification equation 2 return _hash_points(h, gamma, U, V) == c16
```

Game Algorithm Analysis

Analysis of MADAMEFORTUNE Algorithm

Game Algorithm Comparison

Game Type	Randomness Source	Algorithm Complexity	Verifiability	Manipulation Risk
ESLot	VRF → Weight mapping	Medium	Code visible	Medium - Weight distribution
Crash	VRF → Division formula	Low	Formula simple	Low - Mathematical
Dice	VRF → Range check	Low	Formula simple	Low - Mathematical
Jackpot	VRF → Stake selection	High	Complex logic	High - Stake manipulation
Mines	VRF → Grid shuffle	Medium	Shuffle algorithm	Medium - Grid setup
Coinflip	VRF → Binomial	Low	Statistical	Low - Mathematical
Roulette	VRF → Wheel position	Low	Simple mapping	Low - Direct mapping

Algorithm Examples

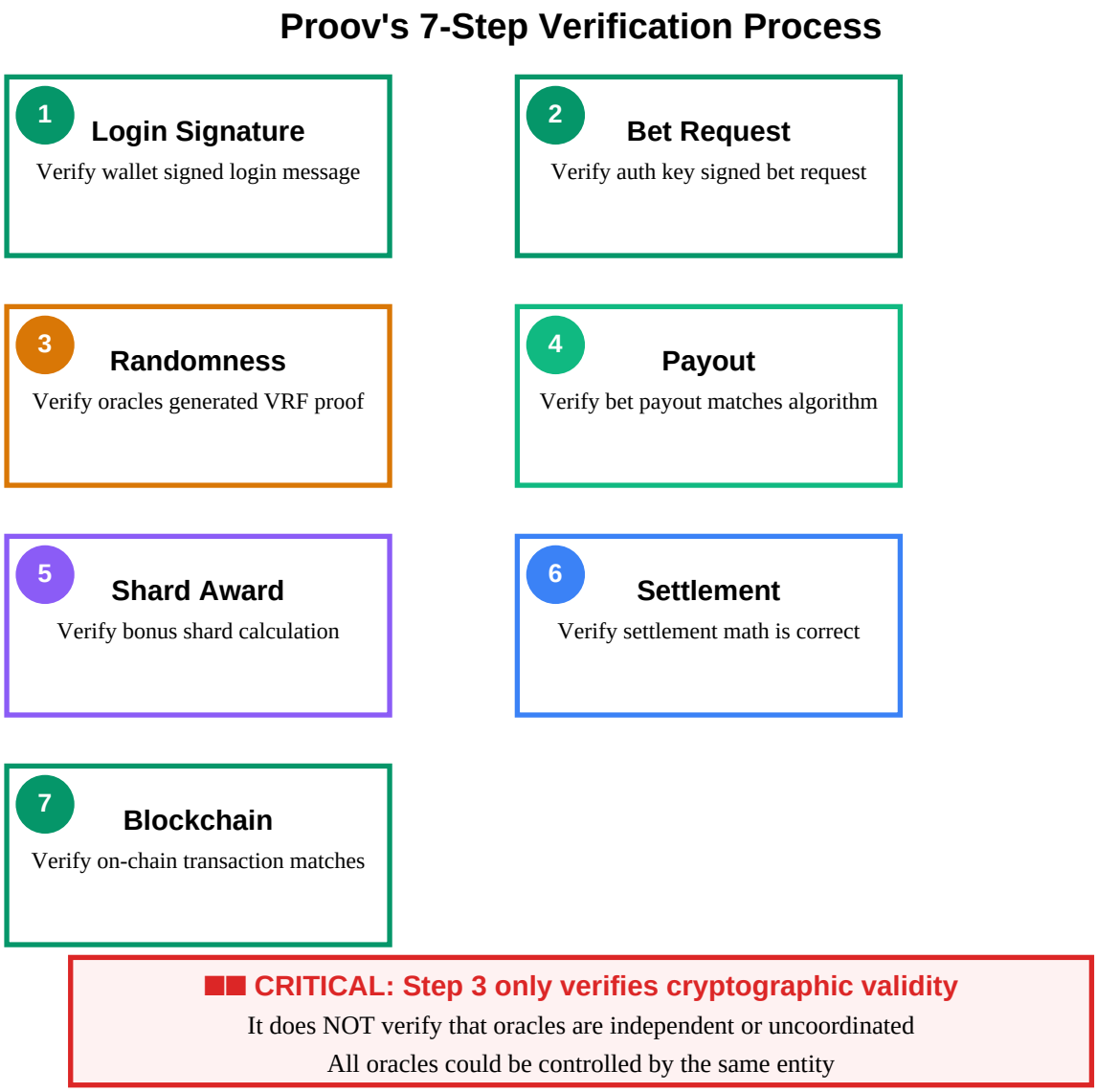
Electronic Slot Implementation:

```
def simulate_eslot(bet: int, distribution: dict, weights: dict, randomness: bytes):
total_weight = sum(weights.values()) outcome = randomness_to_uniform_int(randomness,
total_weight) current_outcome = 0 for (multiplier, weight) in sorted(weights.items()):
current_outcome += weight if current_outcome > outcome: return floor(multiplier /
distribution["bet_multiplier"] * bet)
```

Crash Game Implementation:

```
def simulate_crash(bet: int, edge: float, randomness: bytes, target: float): x =
randomness_to_uniform_float(randomness) crash_multiplier = (1.0 - edge) / max(1e-9, x) if
crash_multiplier > target: return floor(target * bet) else: return 0
```

7-Step Verification Process



Critical Analysis of Trust Assumptions

Oracle Independence: All oracles appear to be controlled by Proov Network. There's no evidence of independent third-party oracles.

Coordination Risk: Nothing prevents the oracles from coordinating their responses to favor the house or specific players.

Key Management: The private keys for all oracles are presumably held by the same organization.

No External Audit: The oracle infrastructure and key management practices are not independently audited.

Centralized Control: Despite using decentralized technology (VRF), the system relies on centralized trust.

Technical Recommendations

For True Decentralization:

Independent Oracles: Use oracles operated by different, unrelated organizations

Threshold Signatures: Require multiple independent oracles to agree on randomness

External Audit: Have independent security firms audit the oracle infrastructure

Open Source: Make all oracle code and verification tools publicly available

Transparent Operations: Publish oracle operation logs and key rotation schedules

Comparison to Industry Standards

Aspect	Proov Network	Chainlink VRF	Licensed Casino
Oracle Count	Multiple (same org)	Decentralized network	Physical/certified
Independence	■ Same company	■ Independent nodes	■ Regulated
Verification	■ Mathematical	■ Mathematical	■ Physical audit
Transparency	■■ Limited	■ Full	■ Regulated
Trust Model	■ Single entity	■ Distributed	■ Government backed