Onchain Bet: A Trustless Protocol for Conditional Bilateral Settlement

Otoshi
https://onchain-bet.org

Abstract

We present a fully decentralized smart contract protocol for trustless conditional bilateral settlement on Ethereum. Two parties lock capital on opposing sides of any verifiable claim, and a mutually agreed human resolver determines the outcome. The protocol requires no external data feed, no oracle network, and no ongoing infrastructure. It supports asymmetric stake ratios, both ETH and ERC-20 tokens, and includes a 30-day safety timeout that guarantees capital recovery if the resolver becomes unresponsive. The contract has no owner, no administrator, no upgrade mechanism, and no pause function.

1. Introduction

In March 2022, a prominent cryptocurrency figure volunteered his personal Ethereum wallet to hold $11 million in stablecoins as escrow for a high-profile conditional agreement between two public figures regarding the future price of a digital asset. Both counterparties deposited their respective stakes to this individual's address. The arrangement functioned because both sides trusted the escrow holder personally.

This event illustrates a gap in decentralized financial infrastructure. Despite the existence of programmable money on public blockchains, no widely adopted mechanism existed for two parties to lock capital on opposing positions and have the outcome settled trustlessly. The participants relied on a single individual's reputation and custody, introducing precisely the categories of risk that blockchain technology was designed to eliminate: private key compromise, social engineering, legal coercion, and the fundamental dependence on human integrity.

Onchain Bet addresses this gap by replacing the trusted intermediary with an immutable smart contract. Both parties deposit their stakes into a contract that neither controls. A designated resolver, agreed upon by both parties before any capital is locked, determines the outcome. The resolver's authority is strictly constrained: they may direct funds to one of the two participants or declare a draw, but no function exists that would allow the resolver to direct funds to their own address or to any address other than the two original participants.

The protocol formalizes a practice as old as human civilization: two people disagree about a future outcome, each puts up capital to back their position, and a trusted adjudicator determines who was correct. The innovation is not the concept but the elimination of custody risk from the adjudicator's role. The adjudicator settles. The code holds.

2. Human Resolvers and External Data Dependencies

A common approach to conditional settlement in smart contracts is to rely on external data feeds, typically provided by oracle networks that relay off-chain information to on-chain contracts. While this approach works for standardized data points such as asset prices or sports scores, it introduces several categories of risk that are incompatible with the goal of permanent, autonomous infrastructure.

Dependency risk. An external data feed is a runtime dependency. If the feed operator ceases operation, raises prices, changes data formats, or experiences downtime during a critical settlement window, the contract may be unable to resolve. For infrastructure intended to operate indefinitely without maintenance, any external dependency is a potential point of failure.

Data scope limitation. External feeds can only provide data that their operators choose to support. A conditional agreement about a private event, a subjective outcome, a personal challenge, or an obscure metric has no corresponding data feed. The space of possible conditional agreements vastly exceeds the space of available data feeds.

Manipulation surface. Any data feed that determines the distribution of locked capital creates an economic incentive for manipulation proportional to the capital at stake. The security of oracle-dependent protocols is bounded by the cost of corrupting the oracle, which may be significantly less than the value of the contracts it secures.

The human resolver model eliminates all three categories of risk. A resolver is a person or entity that both parties agree to trust for the specific conditional agreement at hand. The resolver's authority is constrained by the contract to a single action: calling resolve() with an outcome of PartyA, PartyB, or Draw. No other function is available to the resolver. The contract does not contain any mechanism for the resolver to withdraw funds, modify terms, or redirect capital.

The resolver's economic incentive to act honestly is straightforward: they have no way to profit from dishonesty. An honest resolution maintains their reputation for future resolver engagements. A dishonest resolution gains them nothing, as the funds flow to one of the two parties regardless.

3. Asymmetric Stake Ratios

The protocol supports conditional agreements where the two parties have different levels of conviction about the outcome. Party A specifies both their own stake and the stake required from Party B. These amounts need not be equal.

If Party A deposits 1 ETH and requires Party B to deposit 3 ETH, the total locked capital is 4 ETH. The prevailing party receives the entire pool minus the protocol fee. The asymmetric stakes express the implied probability assessment of each party: Party A believes the outcome is likely enough to accept a smaller potential gain relative to their risk, while Party B believes it is unlikely enough to demand a larger potential gain.

The contract is agnostic to the interpretation of stake ratios. It enforces only the mechanical rule: both parties must deposit the specified amounts, and the prevailing party receives the full pool. Any ratio is supported, from 1:1 to arbitrarily large asymmetries.

4. Protocol State Machine

Each conditional agreement follows a deterministic state machine with five states: Open, Matched, Resolved, Cancelled, and Expired.

4.1 Open

Party A creates the agreement by depositing their stake and specifying parameters: the required counterparty stake, a counterparty address (or zero address to allow any counterparty), the resolver address, a deadline timestamp, and a terms hash. The agreement is now Open and awaiting a match.

4.2 Matched

Party B reviews the terms, resolver, and stake requirements, then deposits their stake. The agreement transitions to Matched. Both stakes are locked. Neither party can withdraw unilaterally. The resolver cannot act until the deadline passes.

4.3 Resolved

After the deadline, the resolver calls resolve() with one of three outcomes: Party A prevails, Party B prevails, or Draw. On a decisive outcome, the prevailing party receives the full pool minus the protocol fee. On a draw, both parties receive their own stakes back minus a reduced fee. The agreement transitions to Resolved and is final.

The resolve function enforces a time-bounded resolution window: the resolver may only act after the deadline and before the deadline plus 30 days. This prevents both premature resolution (before the outcome is known) and indefinitely delayed resolution (which would lock capital permanently).

4.4 Cancelled

If Party B never matches, Party A may call cancelBet() to reclaim their stake. No protocol fee is charged on unmatched cancellations.

4.5 Expired

If the resolver does not act within 30 days after the deadline, both parties may independently reclaim their own stakes by calling reclaimExpired(). Per-party claim flags prevent double-claiming. No protocol fee is charged on expired reclaims. This state ensures that the worst-case outcome for any participant is the return of their original capital, never the permanent loss of funds.

5. The 30-Day Safety Mechanism

The 30-day timeout serves as the protocol's fundamental safety guarantee. It establishes an invariant: no capital can be locked in the contract permanently, regardless of the behavior of any participant, including the resolver.

Consider the possible failure modes without a timeout. If the resolver loses their private key, both parties' capital would be locked indefinitely. If the resolver dies, emigrates, or simply loses interest, the same

outcome occurs. If the resolver is legally compelled not to act, the same outcome occurs. Each of these scenarios is plausible over the intended operational lifetime of an immutable contract.

The 30-day duration balances two competing concerns. A shorter timeout would pressure resolvers to act quickly, potentially before outcomes are clearly determined. A longer timeout would impose an unreasonable delay on participants seeking to recover capital from an unresponsive resolver. Thirty days provides ample time for a diligent resolver to act while bounding the maximum delay for capital recovery.

6. Token Compatibility

The protocol supports both native ETH and ERC-20 tokens. For ERC-20 deposits, the contract uses a balance-before and balance-after measurement pattern on both the Party A and Party B deposits. This symmetric treatment ensures that tokens implementing transfer taxes or fees are handled consistently for both participants.

7. Security Architecture

All state-changing external functions are protected by OpenZeppelin's ReentrancyGuard. All ERC-20 transfers use SafeERC20. All ETH transfers follow the Checks-Effects-Interactions pattern. Existence guards validate that the agreement ID refers to a created record before any operation.

The resolve function enforces both a lower time bound (block.timestamp must be at or after the deadline) and an upper time bound (block.timestamp must be before deadline plus RESOLVER_TIMEOUT). This dual bound ensures that the state machine transitions are fully deterministic and that the Expired state is reachable only after the resolution window has closed.

8. The Case for Immutability

An upgradeable contract has a strictly larger attack surface than an immutable one. Every upgrade mechanism requires at least one privileged address with authority to modify behavior. This address is a permanent vulnerability: it can be compromised through key theft, social engineering, legal coercion, or insider action.

Governance mechanisms distribute but do not eliminate this risk. A governance token creates a market for protocol control. Time-locked governance adds delay but does not prevent a sufficiently motivated adversary from executing a malicious proposal.

An immutable contract eliminates the entire category of administrative risk. There is no key to steal, no governance to capture, no upgrade to exploit, and no emergency function to abuse. The strongest test of a protocol's decentralization is whether it would continue to function if its creators ceased to exist. This protocol passes that test.

9. Contract

```
Address: 0xA79a10033fafe00EB21EBE08b3C4Eff3FcE338E1
Chain: Ethereum Mainnet (Chain ID: 1)
```

```
Owner: None
Compiler: Solidity ^0.8.24, Optimizer 200 runs
```

10. Conclusion

Onchain Bet demonstrates that conditional bilateral settlement can be conducted with zero custody risk and minimal trust assumptions. The protocol separates three roles that are traditionally conflated: the stakeholders who hold economic exposure, the resolver who holds decision authority, and the contract that holds the funds. No single role has the ability to unilaterally extract value.

The human resolver model, combined with the 30-day safety timeout, creates a system that handles any verifiable outcome without external data dependencies while guaranteeing that capital is never permanently locked. The contract is permanent infrastructure on Ethereum that requires no oracle subscription, no server, and no governance process.

References

[1] EIP-20: Token Standard. Ethereum Improvement Proposals, 2015.

[2] OpenZeppelin Contracts. Security library for Solidity smart contracts.

[3] N. Szabo, "Formalizing and Securing Relationships on Public Networks," First Monday, 1997.

[4] EIP-6963: Multi Injected Provider Discovery. Ethereum Improvement Proposals, 2023.

— Otoshi

https://onchain-bet.org