UNO-P2P

A Cheater-proof Peer-to-Peer implementation of a UNO

Luka Finn Born

March 18, 2026

General Concept

- Implementation of a simple card game similar to UNO
- Mix of different architecture styles:
  - Game Sessions: Peer-to-peer
  - Leaderboard, user management: Individual client server components
- Cheating-proof (except majority attacks)

Game Rules

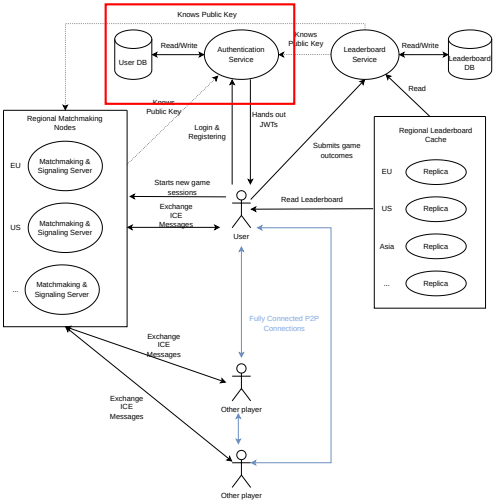
- Each player starts with 7 cards
- One initial card on discard pile
- Goal: be the first to get rid of all your cards
- Each turn:
    1. Optionally draw card(s)
    2. Either:
        - Play a card. The number or the color of the card must match the discard pile
        - Skip the turn

Components - User Management

- Manages user registration & logins
- Hands out JWTs
- Centralized

Dependencies:

- No dependencies

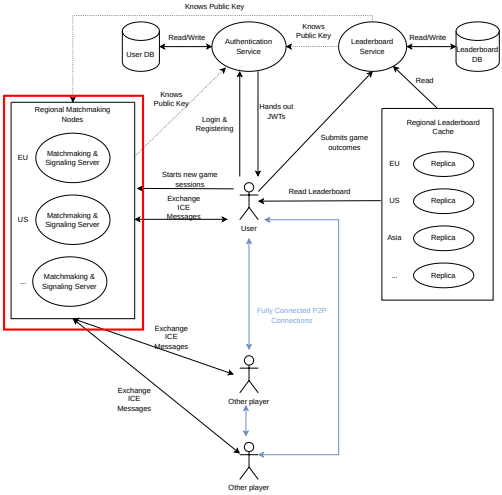


Components - Matchmaking Server

- Manages sessions
- Establishes WebRTC connections between peers
- Multiple regional instances

Dependencies:

- Public key of User management server

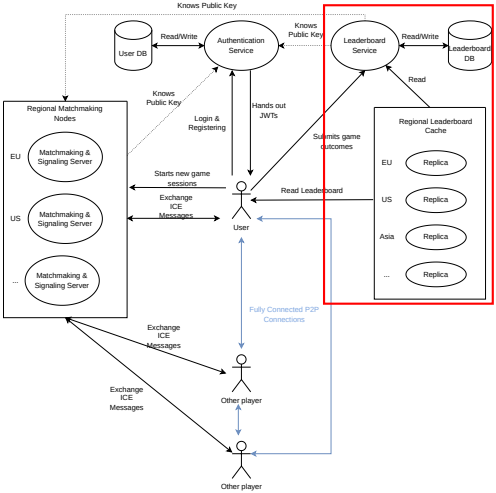


Components - Leaderboard

- Tracks wins of individual users
- Centralized server for writing records
- Distributed caching nodes for read access
- ‘True outcome’ determined by absolute majority

Dependencies:

- Public key of Matchmaking servers
- Public key of User management server

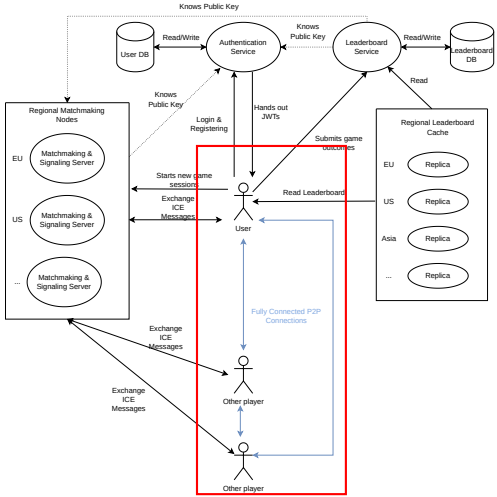


Components - Peer

- Served to the user
- Directly communicates with other components of the system
- Implements game flow & communication during game
- Uses cryptographic protocol to avoid cheating

Dependencies:

- User Management Server
- Matchmaking Servers
- Leaderboard



Peer-to-peer Anti-Cheat

Naive implementation for managing the game state:

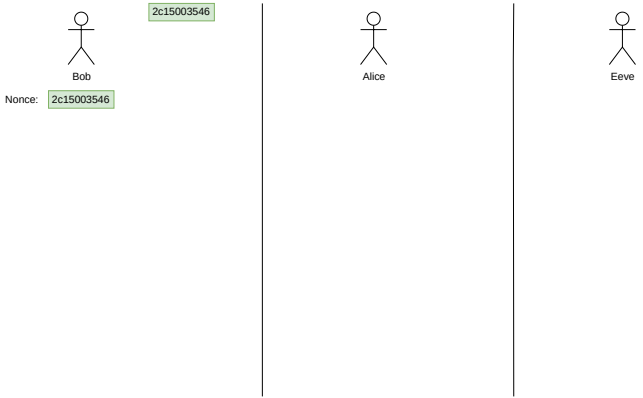
- Select a master/leader peer that knows the whole state
  - Chance of having a malicous master peer
- Each peer knows the whole state
  - Peers can see cards of other peers
- Each peer knows only their state
  - Peers can freely choose cards

Peer-to-peer Anti-Cheat

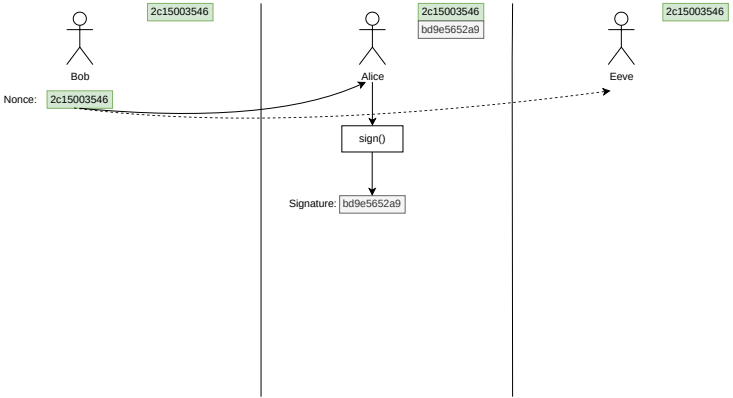
Solution

- Each peer knows only their cards fully
- But: Knows only hash of cards of other players
- Card drawing by chain of cryptographic signatures

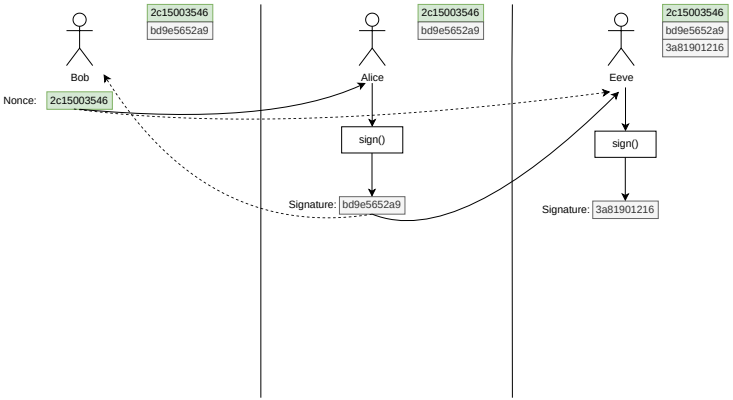
Card Drawing



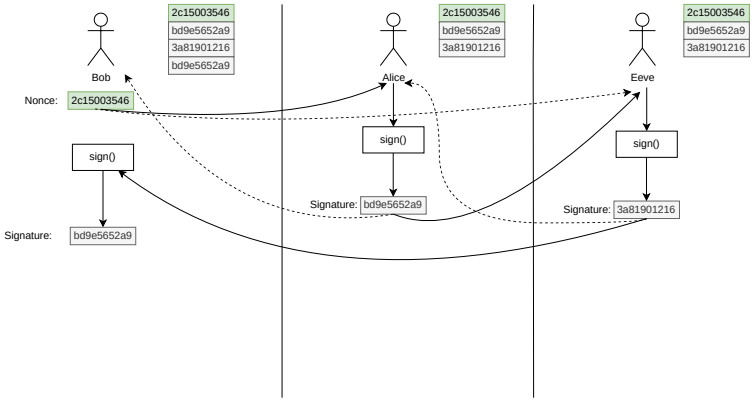
Card Drawing



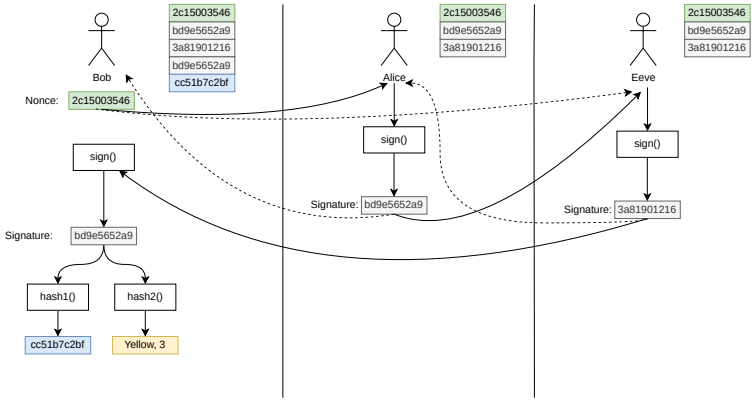
Card Drawing



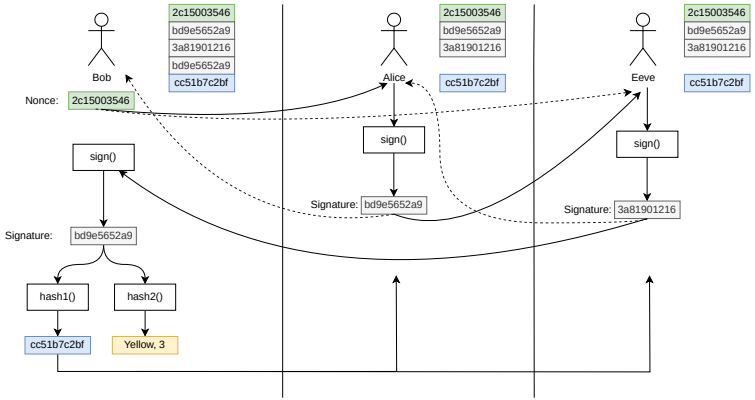
Card Drawing



Card Drawing

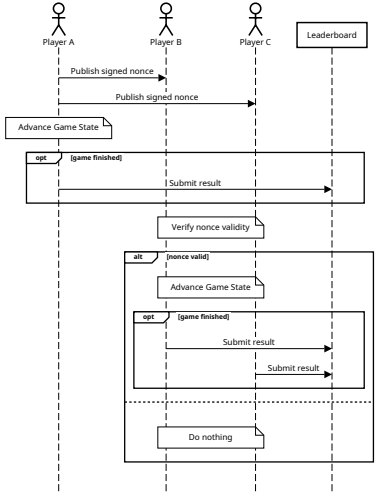


Card Drawing



Play Card

- Player broadcasts final signature of signature chain
- Other players can check that:
  - The signature is valid
  - The hash matches the previously published hash
  - The card can actually be played
- Now each player advances their local game state



Faults

Connection between two players fail:

- If P2P network is still a connected graph
  - Continue
  - Try to reestablish failed edge using P2P network as signaling channel
- If we get multiple disconnected components:
  - Continue with remaining players if majority is still present
  - Abort match otherwise

Faults II

User management server fails:

- Users can no longer login or register
- Already logged in users can still use their JWT

Faults II

User management server fails:

- Users can no longer login or register
- Already logged in users can still use their JWT

Leaderboard server fails:

- Writing new records no longer works
- Reading leaderboard still possible through caching nodes

Faults II

User management server fails:

- Users can no longer login or register
- Already logged in users can still use their JWT

Leaderboard server fails:

- Writing new records no longer works
- Reading leaderboard still possible through caching nodes

Matchmaking server fails:

- We provide multiple distributed instances
- Just use the next-best instance

Technological Details

Client:

- Runs in the browser
- Implemented using TypeScript + React
- Communication via WebRTC data channels

Technological Details

Client:

- Runs in the browser
- Implemented using TypeScript + React
- Communication via WebRTC data channels

Matchmaking server:

- Implemented using Python + FastAPI

Technological Details

Client:

- Runs in the browser
- Implemented using TypeScript + React
- Communication via WebRTC data channels

Matchmaking server:

- Implemented using Python + FastAPI

Leaderboard & User Management server:

- Not yet implemented

Next: Live Demo