Court of Humanity Whitepaper

Justin Greene justin.greene5@gmail.com

February 24, 2026 at 7:50 PM MST

Table of contents

1 Court of Humanity: Private Biometrics, Economic Arbitration, and Social Identity for Sybil Resistance

1.1 Abstract

We propose Court of Humanity, a sybil-resistance identity system coupling private biometric uniqueness with Kleros-like decentralized arbitration and bond-based cryptoeconomics. Users post identity bonds sized to the value at risk; suspicious accounts can be disputed, triggering in-person verification by staked verifiers in a Schelling game. This establishes a quantifiable cost of biometric forgery, enabling developers to set bond sizes ex-ante rather than relying on unverifiable hardware trust or opaque anti-spoofing heuristics. Because effective dispute detection requires observable context about accounts, we introduce Bonafide, a minimal social identity layer where users maintain persistent bonds and public profiles. Bonafide provides the signals — posting patterns, profile authenticity, behavioral consistency — that disputers need to confidently identify sybils across the ecosystem. Downstream applications can require zero-knowledge attestations of Bonafide account standing without learning the user's social identity, preserving cross-application privacy. Biometric uniqueness is computed via secure computation, and verifiers receive only minimal, purpose-scoped disclosures. The result is a general-purpose identity primitive suitable for digital democracy, authentic online identities, and machine-verifiable media provenance.

1.2 How It Works (In Brief)

Fake accounts are a growing crisis. Bots flood social media, scammers impersonate real people, and online votes are trivially gamed. The root cause is that creating a new digital identity is nearly free. Court of Humanity makes it expensive to be fake.

When you join an application that uses Court of Humanity, you put down a small deposit — like a security deposit on an apartment. Your deposit is tied to your biometrics (a scan of your face, for example), and the system privately checks that no one else has already enrolled with the same biometrics. Your raw biometric data is never exposed; all checks happen on encrypted data.

If someone suspects your account is fake, they can challenge you by putting down their own deposit. You then meet a small group of randomly selected verifiers in person, who confirm that you match the biometrics linked to your account. If you are real, you keep your deposit and the challenger loses theirs. If you are fake — or you simply don't show up — you lose your deposit to the challenger. Because verifiers are randomly chosen and put up their own stakes, no single party can corrupt the process.

To make challenges effective, the system needs a way for people to spot fakes in the first place. That is the role of Bonafide, a basic social identity layer where every account holder maintains a public profile and a standing deposit. Posting patterns, profile authenticity, and account age give observers the context they need to decide whether an account looks suspicious enough to challenge. Other applications can require proof that a user holds a Bonafide account in good standing, without learning which account it is, so your social identity stays private across apps.

The result: faking an identity has a clear, calculable cost, applications can size their deposits to the value they need to protect, and the entire system runs without trusting any single company, device, or government.

1.3 Introduction and Motivation

1.3.1 The Problem

In digital systems such as social media or blockchain applications, digital identity is a fundamental component. In such systems, the ability for an adversary to cheaply create multiple identities, known as sybil attacks, often leads to major problems and vulnerabilities. As we will see, many applications are undermined by sybil attacks, where enforcing 1-person-1-account would be hugely beneficial.

1.3.1.1 Social Media Bots

These days, social media is riddled with fake bot accounts. These accounts are used for spam, scams, spreading propaganda, and other nefarious activities. It's fairly easy for someone to create thousands of social media accounts to spread their messages, crowding out posts from real users. Authoritarian governments make heavy use of bots to spread their propaganda and suppress dissent, often running so-called astroturfing campaigns to make it appear as if a large number of people support a particular viewpoint. Some of these campaigns have been fairly successful at getting large swaths of people believing false information. Users are increasingly frustrated with the prevalence of bots and loss of legitimate human interaction.

1.3.1.2 Fake Media

In addition to astroturfing campaigns, fake videos, photos, and audio are increasingly being used to spread false information. This is often done by using generative AI. This content is increasingly indistinguishable from real media, and can be used to spread false information or to manipulate public opinion. There are even ongoing scams where people receive calls from a scammer pretending to be a relative or friend in need of money, using AI to generate the voice of the relative or friend. This increases the importance of media provenance, or the ability to tell from whom the content originated.

On centralized platforms, provenance is made difficult by sybil attacks. Scammers can create many accounts to spread fake content, or fake accounts that look similar to real ones. Fake videos of public officials can circulate widely and cause chaos. Users must ensure they are interacting with the real accounts and not fake ones that look similar. Furthermore, centralized platforms must be trusted not to tamper with the content or the accounts.

1.3.1.3 Online Polling

Online polling is a common way to collect public opinion. However, a sybil attack can be used to manipulate the outcome of a poll by creating many fake accounts to vote multiple times. This can be used to create fake results or to influence the outcome of a poll. Additionally, this makes it difficult to implement any kind of digital democracy.

1.3.1.4 Online Reputation

Online reputation is a foundational element for trust in online marketplaces and communities. For example, on platforms like Ebay, users rely heavily on seller ratings and reviews to determine whether a seller is trustworthy. Buyers are more likely to purchase from sellers with many positive reviews and a strong track record. However, sybil attacks can undermine this system: a malicious actor can create many fake accounts to generate fraudulent sales and reviews, artificially inflating their reputation. These fake sellers may appear highly trustworthy, but in reality, they can use their manufactured credibility to scam unsuspecting buyers. As a result, the effectiveness of reputation mechanisms diminishes, making it difficult for honest users to distinguish between genuine and deceptive actors.

1.3.2 Previous Solutions

Several solutions to sybil attacks have been proposed, with some currently in use. However, we believe that all of these solutions are either unscalable or vulnerable to generative AI.

1.3.2.1 KYC

Centralized systems mainly use government-issued IDs or Know Your Customer (KYC) processes as a means to enforce 1-person-1-account. These processes involve examining documents for forgery, correlating with other data sources, and ensuring personal data is unique. However, forgery detection is not perfect and increasingly less secure with generative AI advancements. Third-party data sources are also vulnerable to forgery and manipulation. Verifying IDs with a government is typically not an option, because if governments allowed this then adversaries could probe the system to learn private information about individuals. Additionally, government-issued IDs are not immune to misuse, as corrupt government officials could create valid but fake identities, or government systems could be hacked. KYC processes require the personal data to be stored, which has led to the leaking of personal data from security breaches of these centralized services.

1.3.2.2 zk-KYC (Passport Chip Proofs)

A newer class of identity systems uses zero-knowledge proofs over digitally signed data stored on passport NFC chips. Over 150 countries now issue e-Passports conforming to the ICAO Doc 9303 standard, embedding biographic and biometric data on a contactless chip. The issuing country's document signer digitally signs this data, allowing any reader to verify authenticity against the country's public key. zk-KYC systems have users tap their passport on a phone, read the signed chip data, and generate a zero-knowledge proof of its validity — proving, for instance, that the user holds a passport signed by a recognized country without revealing the underlying personal data.

While this approach is attractive for its privacy properties, it has fundamental trust model weaknesses that make it unsuitable as a foundation for sybil resistance:

1-of-~200 multisig trust model. The system's security reduces to a 1-of-N multisig where N is the number of entities with access to a country document signing key. Any single government — or any corrupt employee at a passport-issuing agency with access to the private key material — can fabricate passports with validly signed chip data. These forged identities are indistinguishable from legitimate ones to any consuming application. With roughly 200 countries issuing e-Passports, the attack surface is vast: a single compromised key in any jurisdiction is sufficient to mint unlimited valid identities.

Government gatekeeping of digital identity. Because identity derives from a state-issued document, governments retain the power to deny individuals access to online identity. Stateless persons, refugees, and citizens of countries that do not issue e-Passports are excluded entirely. Governments could also revoke or refuse to renew passports to selectively deny dissidents or targeted groups participation in systems built on this primitive.

Centralized root of trust via ICAO. The International Civil Aviation Organization (ICAO), a United Nations agency, maintains the ICAO Public Key Directory (PKD) — the master list of Country Signing CA certificates that consuming applications use to determine which signing keys to trust. Corruption of this directory, whether through institutional compromise, coercion of ICAO staff, or state-level pressure, could result in malicious signing keys being added to the trusted set. Applications relying on the PKD would then accept forged passports signed by the attacker's key as legitimate, with no mechanism to detect or dispute the fraud after the fact.

1.3.2.3 Social Graph Analysis

In social networks, the graph of user connections (IE friends or followers) often contains useful information about the user. There is a tendency for real users to be more well-connected in the graph compared to fake users which typically have fewer or concentrated connections. This can be used to statistically detect fake users by analyzing the user's social graph. However, this method is still vulnerable to sybil attacks, as a malicious actor can create many fake accounts and connect them to each other to create a fake social graph. Projects like BrightID attempt to reduce spoofed social graphs by inserting trusted seed groups from which connections have higher value. However, this ultimately moves the goalpost and requires trusting managers of these seed groups to not create connections to fake accounts. BrightID often has video meetings for new users to get a connection from a seed group member, but this will eventually be vulnerable to live AI generated videos. If BrightID moved to in-person meetups, it would be quite difficult to scale this to billions of users around the world. Additionally, the social graph analysis calculations become very slow to compute as the network grows, especially in a privacy preserving way.

1.3.2.4 Biometric Uniqueness

Biometrics are now increasingly used as additional security measures to authenticate identity. Biometrics are constructed by converting a user's unique physical features into digital data. For example, a facial biometric might look at the distance between the user's eyes, the width of their nose, the shape of their ears, and the color of their skin. Typically hundreds if not thousands of features are used. These features are then converted into a digital representation (vector) that can be stored and compared to other digital representations. Centralized services often require users to submit biometrics to create an account. These biometrics may be checked for uniqueness to ensure the user is not using the same biometrics for multiple accounts and/or used for "liveness" checks to ensure the user is real.

Enforcing users to have unique biometrics can be done by checking new users' biometrics against existing users. When a user signs up, they can submit biometrics that will be checked against the biometrics of all existing users. If the new user's biometrics match with any existing user's biometrics, then the new user is not allowed to sign up (or allowed up to some threshold of matches). This can be done by measuring the "distance" or similarity between the biometric vectors. If two vectors are within some similarity threshold, then they are considered a match. While biometrics are prone to false positive and false negative errors, with some tolerance and tuning it's a reasonable way to compute uniqueness. However, if an attacker is able to submit fake biometrics, then they can create unique-looking biometrics for all their sybil accounts.

1.3.2.5 Decentralized Biometric Systems

Centralized biometric services come with privacy and security risks, as the data is vulnerable to leaks, misuse, and security breaches. Decentralized biometric uniqueness systems have been proposed to address some of the risks of centralized identity systems while providing sybil attack resistance to integrated applications. Decentralized systems can use secure computation methods to prevent anyone but the user having access to the raw data, improving privacy over centralized systems. Worldcoin is one such project that uses special biometric devices called Orbs to scan users' unique iris patterns to ensure 1-person-1-account. These Orbs contain a private key that signs data from the scanner and posts it to a blockchain. This is intended to ensure only unique biometrics from trusted Orbs can be used, preventing spoofed unique data. The trusted public keys of the Orbs are published by a multi-signature scheme involving a small group of Worldcoin employees and advisors, where a majority must sign off on each list of trusted Orb keys. If this multi-signature scheme is compromised, then an attacker can create infinite identities with unique spoofed biometrics that would appear to be from trusted Orbs. Additionally, with thousands of Orbs deployed around the world, it would not be difficult for an attacker to gain physical access to an Orb, and thus Worldcoin's trust model depends on physical security, which is not a scalable solution. A highly motivated and well funded attacker could extract the private key from an Orb, feed spoofed data into the scanner, or alter the machine code. With the immense profit sybil attacks can generate, which could be on the order of tens of billions of dollars for global applications, multi-signature schemes and physical security are likely to be compromised.

1.3.3 Preventing Forgery

Without trusted multi-sigs or liveness algorithms, how can we prevent forged biometrics? Unless you take everyone's biometrics with your own device, how can you trust any biometrics submitted to a biometric uniqueness system were not faked? Without assurances here, we can't use biometric uniqueness to enforce 1-person-1-account, and sybil attacks remain a problem. Various methods have been proposed to determine whether an account is a real human but are increasingly vulnerable.

Some platforms have new users undergo automated intelligence tests like CAPTCHAs that involve image recognition tasks or complex puzzles that are difficult for machines to solve. One blockchain project, Idena, has users periodically solve puzzles where images must be arranged in a narrative order. However, given the rapid advancement of AI both in image recognition and reasoning, these tests are unlikely to be reliable in the long term or against a well funded adversary.

Other use "liveness" tests where the user may be asked to make a certain gesture or facial expression, then machine learning models predict whether the video of the user is real or generated. But again, this involves a cat-and-mouse game between liveness models and generative AI models, which at some point will be good enough to fool even the most advanced liveness models.

1.3.4 In-Person Meetups

Ultimately, the gold standard for proving humanity is meeting in person. By meeting in person, a user could prove to you with high likelihood the biometrics they submitted to the uniqueness system are real, by scanning their biometrics with trusted hardware (eg, your device) and comparing it to what is tied to their account. However, in-person meetups are difficult to scale. Every user would have to meet every other user, or have special trusted users who vouch for others.

1.3.5 Bonded Verifiers

Instead of meeting someone for yourself, or trusting the word of someone else, we could use special bonded verifiers who attest to the humanity of a user. These users have stake that they can lose if they are found to be lying. They could meet new users in person and verify their biometrics. This creates an economic cost to forging biometrics, as forgers would either need to bribe verifiers to approve their accounts or be a corrupt verifier themselves. Corrupt verifiers are eventually exposed and lose their stake, creating an economic cost to forging biometrics. By knowing the cost, applications can reason about how much value they expose to sybil attacks and what is safe.

1.3.6 Dispute-based Meetups

Instead of having all users meet in person, we can use a dispute-based system, where only users who are suspected of not being real will be required to meet in person with verifiers. In most applications, sybil attacks cause observable abnormalities such as repetitive posting of similar content or performing malicious actions. We can use this context to have only a small fraction of users ever needing to meet in person.

1.3.7 Identity Bonds

In a dispute-based system, we need to incentivize users to be on the lookout for suspicious activity and raise disputes. We can provide this incentive and further increase costs to forgery by requiring users to post bonds to their identity/account which they can lose if they fail to pass an in-person verification. Applications could set the bond size based on the value at risk of the action and the likelihood of a sybil attack. For example, a social media app could require a \$5 bond to sign up. If the user makes suspicious posts, someone can dispute the account which will require the user to meet in person with a verifier. If the user fails to show up or does not pass the verification, they lose their bond to the disputer. However, to discourage frivolous disputes, the disputer must also post a bond which they will lose to the disputed user if they are wrong.

1.3.8 The Detection Problem

The bond-and-dispute mechanism depends on the detection probability D: the likelihood that a sybil account is identified and successfully disputed. As the bond sizing analysis will show, low D requires impractically high bond sizes, while high D allows modest bonds that make the system usable.

D depends on observers having enough context about accounts to confidently identify sybils and risk their own dispute bonds. In many application contexts, this context is scarce. An airdrop claimant provides almost no observable signal. A DAO voter is hard to distinguish from a legitimate participant unless the proposal is blatantly malicious — and sophisticated attackers target close votes where they need few accounts and their behavior blends in.

Social media provides the richest detection surface: profile authenticity, posting patterns, account age, engagement behavior, and social graph structure are all signals human observers are naturally skilled at evaluating. This motivates a social identity layer as a foundational component of the system rather than merely another application. We introduce Bonafide, a minimal social identity application launched alongside Court of Humanity, where users post persistent bonds and maintain public profiles. Downstream applications can require attestations of Bonafide account standing to inherit its high detection probability. The full Bonafide design is detailed in a dedicated section below.

1.3.9 Verifier Honesty

To incentivize verifiers to be honest, there must be a significant chance they are caught lying. Verifiers can lie by either approving invalid biometrics or refusing to approve a valid user. To incentivize honesty, verifiers are drawn randomly from a pool and can be appealed to larger groups of verifiers in an escalating court system. Verifiers who vote dishonestly have their stake given to the honest verifiers. We use a game-theoretic approach similar to the decentralized arbitration system of Kleros ("Kleros Yellowpaper," n.d.) to enforce verifier honesty.

Glossary of Key Terms

The following terms appear frequently throughout this paper.

- **Sybil attack**: An attack where a single adversary creates many fake identities to gain disproportionate influence in a system. Named after the subject of a book about a person with multiple personality disorder.
- **Identity bond**: A monetary deposit a user posts as a guarantee that their account is controlled by a real, unique human. Lost if the user fails an in-person verification challenge.
- **Dispute**: A formal challenge raised against an account suspected of being fake. The challenger posts their own deposit and, if the account is found to be fraudulent, collects the account holder's bond.
- **Verifier**: A randomly selected participant who stakes tokens and meets a disputed user in person to confirm their identity. Verifiers are rewarded for honest judgments and penalized for dishonest ones.
- **Schelling point (focal point)**: A game theory concept: when people must coordinate without communication, they tend to converge on the answer that seems most obvious. In Court of Humanity, verifiers independently vote on whether a user is real; because the truth is usually obvious in person, honest voting is the natural coordination point.
- **Bonafide**: Court of Humanity's built-in social identity layer. Users maintain public profiles and persistent bonds, giving observers the context they need to spot fake accounts.
- **Attestation**: A cryptographic proof that a user meets certain criteria (e.g., "account older than 3 months with no failed disputes") without revealing which specific account they control.
- **Zero-knowledge proof (ZKP)**: A cryptographic technique that lets someone prove a statement is true without revealing the underlying data. Used in Court of Humanity so that applications can verify a user's standing without learning their identity.
- **Fully Homomorphic Encryption (FHE)**: A form of encryption that allows computations to be performed on encrypted data without ever decrypting it. Court of Humanity uses FHE so that biometric comparisons happen entirely on encrypted data — the system never sees raw biometrics.
- **fhevm**: A blockchain virtual machine that supports FHE operations, allowing smart contracts to work with encrypted values. Court of Humanity uses Zama's fhevm.

- **Locality Sensitive Hashing (LSH)**: A technique for efficiently finding approximate nearest neighbors in large datasets. Used to quickly narrow down which enrolled biometrics might be similar to a new enrollment, without revealing the biometrics themselves.
- **Layer 1 (L1)**: The base blockchain (e.g., Ethereum) where the court's staking, dispute, and fork logic lives.
- **Layer 2 (L2)**: A secondary blockchain that inherits security from the L1 but offers faster and cheaper transactions. Applications and user bonds live on court L2 chains.
- **Forking**: When a court is captured by a malicious majority, the community can create a copy of the entire system — court, applications, balances — that excludes the attacker. The attacker's version becomes worthless because no one uses it.
- **DAO (Decentralized Autonomous Organization)**: An organization governed by rules encoded in smart contracts and controlled by its token holders, rather than by a central authority.
- **Federation (DAO-of-DAOs)**: A meta-governance structure where multiple democracy DAOs on separate court L2s each cast a weighted vote on cross-jurisdictional proposals, aggregated by a meta-DAO contract on L1.
- **Dispute-Dependent Action (DDA)**: A high-stakes operation (e.g., a DAO treasury transfer) that remains pending until a dispute window closes, giving observers time to challenge suspicious participants before the action executes.
- **Cosine similarity**: A measure of how similar two vectors are, based on the angle between them. Used to compare biometric feature vectors; a high cosine similarity between two enrollments suggests they may belong to the same person.

1.4 Decentralized Arbitration via Kleros

Court of Humanity relies on decentralized arbitration as a foundational primitive. The identity dispute game — where verifiers are drawn, votes are cast, and stakes are redistributed — is built on the mechanism design of Kleros ("Kleros Yellowpaper," n.d.), a decentralized court protocol implemented on Ethereum. This section summarizes the oracle problem Kleros addresses, the game-theoretic core of its mechanism, and why it serves as the right foundation for identity arbitration.

1.4.1 The Oracle Problem

Smart contracts can automatically execute programmed logic, but they cannot make subjective judgments or access information from outside the blockchain. When a dispute arises — whether a freelancer delivered work as specified, whether an insurance event occurred, or whether a submitted biometric belongs to a real person — someone must provide the answer. This is the *oracle problem*: how to get truthful information about the real world onto a blockchain, where the information can be acted upon by smart contracts.

Traditional dispute resolution is too slow, too expensive, and too centralized for a decentralized global economy. A single arbitration case in international commerce can cost tens of thousands of dollars and take months or years. For the kinds of frequent, low-value disputes that arise in blockchain applications — a $200 freelance contract, a $5 identity bond — traditional mechanisms are entirely impractical. What is needed is a fast, inexpensive, and decentralized mechanism that produces reliable judgments without requiring trust in any single party.

1.4.2 Schelling Points and Incentivized Truth-Telling

Kleros addresses the oracle problem using a game-theoretic mechanism rooted in the concept of *Schelling points* (focal points). Game theorist Thomas Schelling observed that when agents must coordinate without communication, they tend to converge on answers that seem natural or obvious — the "focal point." If asked to meet a stranger in New York City with no further instructions, most people independently choose noon at Grand Central Terminal. There is nothing inherently optimal about this choice; it succeeds because people expect others to choose it.

Vitalik Buterin applied this insight to design the *SchellingCoin* mechanism for decentralized oracles. Agents are asked a question with an objectively observable answer (e.g., "Did it rain in Paris this morning?"). Each agent votes by secret ballot. After all votes are revealed, agents who voted with the majority are rewarded, and those who voted against the majority are penalized. When the question has a clear true answer, truth becomes the Schelling point. Each agent votes truthfully because they expect others to vote truthfully, because they expect others to expect truthful votes, and so on. Honesty is the equilibrium.

This mechanism creates an *economic cost of untruthful data*. Anyone providing false information expects to lose their stake to the honest majority. The stronger the consensus around the true answer, the more expensive it becomes to push through a false one.

1.4.3 Kleros Mechanism

Kleros builds on the SchellingCoin foundation with several mechanisms that make it practical for real-world dispute resolution.

Staking and juror selection. Kleros uses a dedicated token (PNK) that participants stake to become eligible jurors. The probability of being drawn for a specific dispute is proportional to the amount staked. Staking serves three functions: it provides sybil resistance (an attacker must acquire proportional stake to gain proportional influence), it creates economic skin-in-the-game that incentivizes honest voting, and it enables the forking mechanism described below.

Court specialization. Kleros organizes courts in a tree structure rooted at a General Court. Specialized subcourts handle domain-specific disputes — software quality, insurance claims, content moderation — allowing jurors to self-select into areas where they have expertise. When appealed beyond a subcourt's capacity, cases escalate to the parent court with a broader juror pool.

Commit-reveal voting. Jurors commit to their votes via cryptographic hashes before revealing them, preventing jurors from simply copying others' votes. This preserves the independent-judgment property that makes the Schelling mechanism effective.

Token redistribution. After voting concludes, jurors whose votes are *coherent* with the final outcome receive a share of the stakes lost by *incoherent* jurors (those who voted against the outcome), plus a share of arbitration fees. This is the core Schelling game: honest jurors profit in expectation, while dishonest or lazy jurors lose their deposits.

Appeals. If a party believes a ruling is incorrect, they can appeal. Each appeal round draws a new panel with roughly double the previous number of jurors plus one, and appeal fees rise correspondingly. This makes bribery exponentially more expensive at each round: an attacker who bribes a 3-juror panel must be prepared to bribe a 7-juror panel, then 15, then 31, and so on. In practice, most disputes settle in early rounds; the mere possibility of appeals deters dishonest behavior. Appeals can also be crowd-funded, allowing third parties who believe a ruling is unjust to finance further review in exchange for a share of the winning side's rewards.

Forking. If an attacker acquires a majority of staked tokens and controls the final appeal round, Kleros provides a last-resort defense: the community can *fork* the system, creating a parallel version in which the attacker's tokens are excluded and the honest ruling is restored. The attacker's tokens on the captured fork become economically worthless, since no one will use a court known to be compromised. This ensures that the cost of a 51% attack includes not just acquiring the tokens but also absorbing their total loss of value when the community migrates to the honest fork.

1.4.4 Why Court of Humanity Builds on Kleros

Court of Humanity's identity dispute game maps directly onto the Kleros architecture. Where Kleros jurors evaluate evidence about contract performance or factual questions, CoH verifiers evaluate biometric evidence about identity claims. The same game-theoretic guarantees apply:

- **Verifier honesty through Schelling incentives.** Verifiers who vote coherently with the final outcome are rewarded; incoherent verifiers lose stake. Since an in-person biometric check has a clear true answer — the person either matches the enrolled biometrics or does not — honesty is the strong Schelling point.
- **Escalating defense through appeals.** A corrupted initial panel of 3 verifiers can be appealed to larger panels, making it prohibitively expensive to bribe enough verifiers to sustain a false ruling through all rounds.
- **Ultimate defense through forking.** If an attacker captures a majority of a court's staked tokens, the community can fork, isolating the attacker and restoring honest dispute outcomes. Court of Humanity extends Kleros's forking concept with child L2 blockchains, allowing application state — including user bonds and balances — to fork alongside the court, preventing attackers from extracting stolen value.
- **Geographic court hierarchy.** Kleros's court tree maps naturally to CoH's geographic jurisdiction model. Local subcourts handle initial disputes near users' locations; appeals escalate to regional or global courts with larger verifier pools and higher security budgets.

The adaptations Court of Humanity makes to the base Kleros design — in-person biometric verification instead of document review, geographic jurisdiction binding, child L2 forking, and bond-based economics — are detailed in the Technical Design sections that follow.

1.5 Technical Design

1.5.1 System Overview

Court of Humanity targets three design goals:

- Privacy by default: biometric uniqueness is computed without disclosing raw biometrics, with deletion/expiry and purpose-limited use.
- Quantifiable cost of forgery: a bond-and-arbitration mechanism sets an explicit economic deterrent that apps can size against their value at risk.
- Composability: simple interfaces for applications to require bonds, query scoped uniqueness, and hook disputes into critical workflows.

Our core observation: while no biometric system can be perfectly unforgeable, we can make forgery economically irrational by combining (a) private uniqueness checks to limit parallel identities and (b) a credible in-person verification game that slashes attackers when they matter.

1.5.1.1 Why a Censorship-Resistant Ledger

The security of the bond-and-dispute mechanism depends on every participant being able to submit transactions within bounded time windows. Users must respond to disputes, disputers must open cases before bonds are withdrawn, verifiers must submit votes before round deadlines, and communities must be able to execute forks when courts are captured. If any actor can be selectively prevented from performing these actions, the system's economic guarantees collapse.

Consider what happens when a single entity controls the underlying database or transaction ordering. A dispute is opened against a user, but the operator delays or drops the user's response transaction. The user is unable to appear, triggering a procedural default that slashes their bond — not because they were a sybil, but because they were censored. The attacker profits from the slash while the victim had no recourse. Worse, a colluding operator could systematically target high-value bonds, dispute them through allied accounts, and censor the victims' responses to collect slashes at scale.

The threat extends beyond individual loss. An operator who controls transaction inclusion can selectively block honest verifiers from submitting votes, manipulating dispute outcomes without needing to acquire a majority of staked tokens. They can prevent fork transactions from being processed, neutralizing the last-resort defense against court capture. They can front-run bond withdrawals or selectively delay dispute-opening transactions past withdrawal deadlines, allowing sybil operators to escape with their bonds intact. In each case, censorship converts the system's economic security into a tool for extraction.

A traditional centralized database places this power in the hands of a single operator. Even a federated system with a small set of operators is vulnerable to collusion or coercion. The system therefore requires a ledger where no single party — and no small coalition — can deny transaction inclusion. This is the property that public blockchains are designed to provide: a decentralized network of independent validators, each with the ability to include transactions, ensures that censorship requires capturing an impractical fraction of the network. Forced-inclusion mechanisms on Layer 2 chains extend this guarantee from the base layer to the application layer, ensuring that even a misbehaving L2 sequencer cannot indefinitely block a user's transactions.

This censorship resistance is not incidental to the design — it is foundational. The bond sizing formula assumes users can defend themselves when disputed. The forking mechanism assumes communities can coordinate on-chain when courts are captured. The procedural default rules assume that no-shows reflect genuine absence rather than suppressed transactions. Without a credibly censorship-resistant ledger, each of these assumptions fails, and the system reduces to one where the database operator holds unilateral power over users' funds and identities.

Actors:

- Users: enroll with private biometrics, bond to use applications, and act. Typical use will involve application-specific sub-accounts for privacy and privacy-preserving funds transfers to avoid linking identities across applications.
- Application developers: define policies (bond sizes, thresholds) and integrate dispute hooks.
- Verifiers: staked participants who adjudicate disputes via a Schelling game, voting with stake.
- Disputers/appealers: trigger and escalate disputes by posting fees/bonds.
- Courts: Kleros-like organizations associated with specific geographies that assign verifiers to disputes and appeals.
- Asset issuers: Authorities that issue assets external to the court (ex Circle).

Assets:

- Identity bonds: posted by users per app/purpose; may be ephemeral (time-limited) or persistent.
- Staking token: secures the court; coherent voters accrue value from incoherent voters; forkable under corruption.
- Dispute fees: paid by disputers/appealers; help fund verifiers and deter frivolous cases.
- External assets (ex USDC): bridged to court L2s for use in applications.

Components:

- Court contracts: Arbitration smart contracts on a Layer 1 blockchain
- Court L2s: Child layer 2 blockchains that fork with the Layer 1 court contracts and state.
- Fork registry: A contract on the Layer 1 blockchain that registers forks and their lineage.
- Vaults: Contracts on the Layer 1 blockchain that hold bridged assets for the court L2s.

- Bonafide: The reference social identity application deployed on court L2s, providing persistent bonds and public profiles as a sybil detection substrate for the ecosystem.
- Applications: Smart contracts on the court L2 blockchain that utilize bonds, disputes, and Bonafide attestations.

Lifecycle:

1. Enrollment: user produces a biometric feature vector locally; the frontend encrypts it to a secure decentralized backend (FHE). Contracts ingest it as an encrypted handle; Only opaque handles are stored on-chain; per-bucket data never appears on-chain.
2. Social identity: user creates a Bonafide profile and posts a persistent identity bond, establishing the public presence that downstream applications can require attestations of.
3. Application bonding: on first use with a downstream application (or per high-stakes action), user posts an additional identity bond per app policy, optionally providing a Bonafide attestation to qualify for reduced bond requirements.
4. Action: the app checks policy by first using Locality Sensitive Hashing (LSH) to privately select a candidate biometric match set, then performing an encrypted fixed-point vector-similarity test against those candidates. For example the policy may require that no candidate's similarity exceeds a threshold τ within the app's scope. Enforcement uses either no-decrypt gating (nullify side-effects when policy fails via FHE.select) or a single-bit decrypt via the decryption oracle to `require(ok)`. Apps can add context-scoped nullifiers to prevent replay within a purpose.
5. Dispute: suspicious accounts/actions are disputable for a window; disputers post a dispute bond. The disputed user presents a short-lived EIP-712 "auth ticket" enabling consented verification without revealing match sets; bond is frozen.
6. Verification: staked verifiers meet the user in person, confirm account control, and run a consent-scoped "does this live scan match the enrolled account?" check using the same division-free cosine test on encrypted vectors. Results are revealed to verifiers via fhevm user-decryption (reencryption to verifier key); minimal information is disclosed.
7. Appeal/fork: losers can escalate rounds; final-round failures can trigger court forking per our forkable architecture.

1.5.2 Humanity Court Design

Humanity Courts are a fundamental building block of the system. Each court or jurisdiction is essentially a modified fork of Kleros with its own token and parameters. The creation of courts is up to the free market to determine the optimal number and distribution of courts. Anyone can start a court by deploying contracts and garnering some initial credibility with an initial token distribution and consuming applications. Courts will be associated with specific geographies in which verifiers and users are required to meet. Users will post bonds in courts bound to their local region. Courts covering larger regions can garner more security but require more potential travel for users and verifiers. Courts have a Decentralized Autonomous Organization (DAO) that governs the court and its parameters. Applications will approve courts they integrate with in order to accept users with bonds in that jurisdiction.

1.5.2.1 Court Organization

People who become verifiers by staking tokens will be required to meet disputed users in certain locations. This could be a general location like a city, where users and verifiers negotiate a meeting time and location. Or it could be a dedicated space for court meetings. DAO governance can be used to set meeting locations and procedures.

Courts can be hierarchically organized with subcourts being associated with specific locations. For example, we could have a Continental US court with a subcourt for each state. A user can post a bond in the California subcourt, where they will need to show up in San Francisco for the initial dispute. If that is appealed, the case moves up to the national court which takes place in New York City where the user and selected verifiers will need to travel. In the early dispute rounds, verifiers are drawn from local subcourts to minimize travel. In the final round, verifiers are drawn from the global pool.

1.5.2.2 Local Courts

While there may be global courts for high security matters, we think most applications will use local courts where users won't have to ever hop on a plane to verify their biometrics. If a court may require significant travel costs for the user, such as a plane ticket, the user may be disincentivized to attend disputes for low value bonds. Ex, many people would not buy a $500 plane ticket to defend a $5 bond. In such a situation, malicious disputers can dispute everyone with low bond values, expecting that most will not show up to later rounds that require significant travel. Thus, large geography courts should probably only be used for high security contexts where bond sizes are significant.

1.5.2.3 Global Court and Treasury Placement

Multi-court applications face a treasury placement choice: either segment assets across court-chain instances or select a single "home" court where the DAO treasury and any global, high-value dispute-dependent actions (DDA) reside. To minimize fragmentation and simplify fork coordination, we recommend bootstrapping with at least one global court that anchors treasuries and protocol-wide decisions, while local courts handle user bond enrollment and routine, jurisdiction-scoped disputes.

Example: a social media app can deploy local court chain instances to accept user bonds in each jurisdiction (improving accessibility and minimizing travel), while keeping the DAO treasury and protocol-level governance on the global court. Routine posting and moderation flows depend on local bonds; treasury moves and other global DDAs execute on the global court.

1.5.2.4 Court Auditing

Users and application owners should be encouraged to audit the court for themselves. The public should be able to contact the disputed user and request a meeting to verify the biometrics. If these results differ from what verifiers voted, then the dispute should be appealed.

In the final round of a dispute, asset issuers, application owners, and high-value users should audit the disputee to determine if there is a 51% attack, and if so, which fork to support. This will require some organization and protocols by the court as this could be a large public event.

1.5.2.5 Governance and Parameters

- η (utilization), fee rates, appeal escalations, dispute windows, and verifier selection parameters are governed by stakers and stakeholders via on-chain proposals.
- Futarchy may be used for proposal voting.
- Upgrades: biometric model and backend upgrades follow opt-in epoching with migration tooling and community audits.
- Safeguards: emergency halts for specific scopes, fork procedures, and transparency reports for dispute statistics and backend audits.

1.5.2.5.1 Futarchy

Futarchy is a mechanism for decision-making that uses a prediction market to determine the outcome of a decision. It is resistant to an entity controlling 51% of tokens to alter court parameters or code in their favor. For this reason it should be the main method of governance in court DAOs for critical decisions.

When a proposal is made, two “conditional markets” are created:

- A market for buying/selling the token if the proposal passes
- A market for buying/selling the token if the proposal fails

If the condition does not happen, then the trades are reverted in that market. A proposal passes if the time-weighted average price of the token in the pass market is significantly (ex, 3%) higher than the time-weighted average price (TWAP) of the fail market. This signals that the market believes the proposal is a net benefit to the value of the token.

1.5.3 Crypto-economic Design

1.5.3.1 Identity Bonds

When a user posts a bond they are essentially committing to a biometric vector/hash that they will have to match in person if they are disputed.

- Purpose: impose a cost on accounts whose behavior triggers disputes; make successful forgeries unprofitable in expectation.
- Sizing: applications set bond sizes relative to their value at risk and their detection likelihood.
- Duration: applications determine when the user can withdraw funds from or return the bond, such as after some challenge period or when the user cancels it.
- Fees: bonds incur fees in the form of an interest rate paid to court staked tokens.
- Accrued interest rate fee is taken out of the bond on a periodic basis.

1.5.3.2 Dispute Game (Kleros-like)

- Randomly drawn verifiers vote with stake; coherent voters (majority by stake) receive the incoherent voters' stakes and dispute fees.
- Appeals: escalations draw more verifiers and increase costs; the final round is large enough to render bribery/capture unprofitable relative to the staked market value.
- Forking: if attacker controls 51% of the voting stake, honest participants can fork into a new court; captured tokens become isolated with little economic value.

1.5.3.3 Bond Sizing Heuristic

The core question for any application using Court of Humanity is: how large should the deposit be? The answer depends on three things: how much value is at stake, how many fake accounts an attacker would need, and how likely it is that the attack gets caught. The intuition is straightforward: if there is a 90% chance that fake accounts are detected and disputed, the attacker loses their deposits 90% of the time, so the deposits only need to be large enough to make the math unfavorable. If detection is less reliable, deposits must be higher to compensate.

The following examples illustrate the tradeoff before presenting the general formula.

Example: DAO voting. A DAO has a $100M treasury. A proposal could redirect 1% ($1M). There are 1,000 voters; an attacker needs 500 fake accounts (simple majority). If observers catch the attack 90% of the time ($D = 0.9$), each fake account needs a bond of only about $222 to make the attack unprofitable in expectation.

- Value at risk (V) = $1,000,000
- Accounts needed ($T \cdot N$) = 500

- Detection probability (D) = 0.9
- Required bond: B ≥ ((1−0.9)/0.9) · (1,000,000/500) ≈ $222

Example: airdrop. A token airdrop gives $20 per unique human. An attacker only needs one fake account per extra claim. If observers catch 80% of fraudulent claims (D = 0.8), a $5 bond per claim makes sybil farming unprofitable.

- Claim value (A) = $20
- Detection probability (D) = 0.8
- Required bond: B ≥ ((1−0.8)/0.8) · 20 = $5

General formula. Let V be the value at risk, N the number of participants whose actions affect the outcome, T the pass threshold fraction (e.g., 0.5 for simple majority), and D the probability that an attack is detected and successfully disputed. An attacker must supply at least T·N accounts, and each account's bond is only forfeited upon detection. To make the attack negative expected value:

$$D \cdot (T \cdot N) \cdot B \geq \ (1 - D) \cdot V \quad \Rightarrow \quad B \geq \frac{1-D}{D} \cdot \frac{V}{T \cdot N}$$

For single-claim scenarios like airdrops, set (T·N = 1) and let (A) be the claim value:

$$B \geq \frac{1-D}{D} \cdot A$$

The detection probability D is the critical lever. Bonafide's social identity layer is designed to maximize D by giving observers rich context for spotting fakes, which in turn keeps bond sizes practical.

Value Accrual

- Juror fees alone underpay security; we add bond fees accruing to stakers.

1.5.4 Dispute Economics and Procedure

This section specifies how to size dispute bonds, draw and stake verifiers, scale appeals, and prevent "no-show" griefing, relying on explicit cryptoeconomic parameters.

In plain terms: When someone challenges a suspicious account, both sides put money on the line. The challenger's required deposit is set high enough to discourage frivolous challenges but low enough that confident observers will still step forward. A small panel of randomly drawn verifiers meets the user in person and votes on whether they are real. If the initial panel gets it wrong, the losing side can appeal, which doubles the panel size and roughly doubles the cost to corrupt it. This escalation makes sustained bribery exponentially expensive at each round. If a challenged user simply refuses to show up, they are penalized immediately — they cannot dodge a penalty by appealing later. The parameters below formalize these dynamics.

1.5.4.1 Notation

- B: total active identity bonds of the disputed account in the app's scope (frozen on dispute)
- Dd: disputer's dispute bond (escrowed on open)
- Fr: total juror fees in round r (Fr = Mr · fr, with fr per-juror)
- Mr: number of verifiers (jurors) drawn in round r (odd)
- smin: minimum stake per drawn verifier; α: max per-vote slash fraction; per-juror at-risk Dj = α·smin
- Sr: total juror stake at risk in round r, Sr = Mr · Dj
- p: disputer's subjective probability the account ultimately fails court
- q: single-juror correctness probability in round r given evidence
- εr: target upper bound on misjudgment probability in round r

1.5.4.2 Disputer Bond Sizing (sets the belief threshold for +EV disputes)

Disputer expected value:

$$\text{EV} \approx p \cdot B - (1-p) \cdot D_d - F_r$$

Break-even belief:

$$p^* = \frac{D_d + F_r}{B + D_d}$$

Let $D_d = \kappa \cdot B$. For $F_r \ll B$:

$$p^* \approx \frac{\kappa}{1+\kappa}$$

Guidance (choose κ by risk class):

- Social/content: $\kappa \in [1.3, 1.8] \Rightarrow p^* \approx 0.56 — 0.64$
- Governance/high-value: $\kappa \in [2.0, 2.5] \Rightarrow p^* \approx 0.67 — 0.71$

Liquidity:

- Allow pooled disputes (multiple disputers fund D_d and F_r).
- Optional partial coverage: a dispute can initially cover fraction c of B with $D_d = \kappa \cdot (cB)$; only cB is at risk until topped up.

Revenue split on slash: route x% of B to stakers (security budget) and (1–x)% to the disputer.

1.5.4.3 Verifier Draw and Stake Sizing (round 1)

Choose the smallest odd M_1 that satisfies both:

A) Statistical correctness:

$$\Pr[\text{majority wrong}] \;=\; \sum\_k = (M_1+1)/2^{M_1} \binom{M_1}{k} (1-q)^k q^{M_1-k} \;\le\; \varepsilon_1$$

Typical targets (in-person verification $q \approx 0.85$–0.9):

- $M_1 = 3 \Rightarrow \varepsilon_1 \approx 6\%$ at $q = 0.85$
- $M_1 = 5 \Rightarrow \varepsilon_1 \approx 1$–$2\%$

B) Anti-bribe economic coverage:

$$S_1 = M_1 \cdot D_j \ge \beta \cdot (B + D_d), \quad D_j = \alpha \cdot s_{\text{min}}$$

Pick $M_1 = \max(\text{stat bound}, \text{econ bound})$ and set f_1 so expected honest juror payoff covers effort (cf. Kleros).

1.5.4.4 Appeals

- Panel growth: $M_{r+1} = 2M_r + 1$
- Fee growth: $f_{r+1} = \lambda \cdot f_r,\ \lambda \in [1.5, 2.0]$
- Crowd-fundable appeals on both sides (loser-pays variant optional)
- Sr scales with Mr to raise sustained bribe costs super-linearly
- Final round may trigger forking per Court architecture

1.5.4.5 Procedural Non-Compliance (No-Show) Rule

To prevent “no-show then appeal” griefing, separate procedure from substance:

- On dispute open, the user must post an appearance assurance $A_u = \sigma \cdot B$ and schedule verification by deadline T_0.
- If the user no-shows or refuses biometrics in round r, that round issues a Procedural Default:
  - Immediate, non-reversible penalty: slash $\rho_r \cdot B$.
  - Distribution: y% to the disputer, z% to round-r jurors, remainder to stakers.
  - The remaining $(1 - \rho_r) \cdot B$ stays frozen for subsequent substantive rounds.
  - The disputer’s D_d remains pending; procedural penalties are never reversed by later evidence.

- To appeal after a procedural default, the user must pre-post A_u (if missing) and next-round fees; prior procedural slashes $\sum \rho_r B$ stand regardless of substantive outcome.

This guarantees early jurors cannot be harmed ex post by later evidence and removes incentives to self-dispute/no-show grief.

1.5.4.6 Anti-Collusion and Auditing

- Commit-reveal with re-commit allowed during vote; penalize pre-revelation via a Process Court.
- "Shadow auditors" are randomly drawn with small stake and a bounty funded from A_u and F_r; they can trigger Process Court if meetup protocol deviates.
- Coherence/slashing:
  - Procedural rounds: judged only on procedural facts of that round; payouts final per round.
  - Substantive rounds: coherence measured against the final substantive outcome (as in Kleros).

1.5.4.7 Default Parameters (per court class)

- Dispute bond multiplier κ: 1.5 (social), 2.0 (governance)
- Round 1: $M_1 = 3$ (social, $\varepsilon_1 \approx 6\%$), $M_1 = 5$ (governance, $\varepsilon_1 \approx 1\text{–}2\%$)
- Stake coverage $\beta = 0.5$ (social), 1.0 (governance)
- Juror slash factor $\alpha = 0.5$ choose $s_m in$ so $D_j = \alpha s_m in$ is meaningful
- Appeal scaling: $\lambda = 2.0$, $M_{r+1} = 2M_r + 1$
- Procedural slashes: $\rho_0 = 0.3$, then +0.1 per missed round capped at 0.6; $\sigma = 0.3$
- Procedural slash distribution: y=60% (disputer), z=20% (jurors of that round), 20% (stakers)
- Protocol cut on substantive slash B: x=20–30% to stakers (rest to disputer)

1.5.4.8 Similarity Parameters (per modality/app)

- Cosine threshold τ for fine similarity: calibrated per model and risk class (example ranges: 0.75–0.85 social; 0.82–0.9 governance), represented as fixed-point `tauQ`.
- Fixed-point format for vectors: e.g., Q1.15; dimension d (e.g., 128–512) chosen per model.
- Candidate limits: max buckets per probe and max candidates per check to cap cost.

These knobs make the disputer's break-even belief explicit ($p^* \approx \kappa/(1+\kappa)$), ensure first-round juries are both statistically and economically robust, escalate defenses with appeals, and remove incentives for timing-based griefing.

1.5.5 Forking

The appeal process in a court is designed to protect disputes against bad verifiers, pulling in more and more verifiers/stake as disputes escalate. But if an attacker controls 51% of the staked tokens in a court, whether acquired, bribery, or collusion, they can resolve disputes in their favor even in the final appeal round. When this happens, anyone can create a copy of the court contracts and state in which tokens that voted for the attacker are excluded and the case resolved honestly. This creates a separate instance of the court, or a fork, which honest tokens can migrate to. The court token of the original court captured by the attacker will be worthless as no one will want to use it. The forked court token will retain value as a legitimate court.

Thus, forking is a last resort against 51% attacks. The mere threat of a fork should discourage 51% attacks most of the time. However, we build on Kleros's concept of forking with child layer 2 blockchains that allow forking application state along with the L1 court contracts. This allows undoing the consequences of 51% attacks and prevents attackers from running away with stolen funds or bonds of legitimate users.

When a dispute reaches a final appeal round, courts should notify applications and users of a potential fork. Ultimately users need to determine what fork to support, so many people will want to attend the final round to verify the disputed user for themselves. In the case of a 51% attack trying to defend a sybil account, anyone attending the final round could see that no one is able to provide biometrics matching the sybil account. Then the community can rally around a fork that rightfully rejects the sybil account. After the final round voting ends, an honest fork is created and users configure their clients to use the new fork.

To bridge assets into a court L2, assets must be deposited in a vault contract that gives the asset issuer control over what L2 fork to honor withdrawals from. With L2 withdrawal delays, asset issuers can prevent a 51% attacker from withdrawing stolen assets because the asset issuer will honor the honest fork where the attack was reverted. For example, if Circle wants USDC to be usable in a court L2, say for bonding, they would create a vault contract for that court and be on the hook for fork choice. Users of these assets would be putting trust in the asset issuer's commitment to honor the honest fork. The native court tokens will always be useable in a trustless manner on the L2 since court tokens on the L2 will point to the new court tokens in a fork.

While anyone can manually deploy a fork of the system at any time, forks that emerge from a final dispute round can be registered in an official fork registry contract to assist coordination around the new contract addresses and provide a notification to blockchain clients about the fork.

1.5.5.1 Fork Flow

Court root (L1): CourtHub per court with staking, disputes, and a fork tree. Publishes checkpoints used by its child chain(s).

Child chain (L2): Apps live here. The child chain state root is anchored to the court. On fork, the child chain is duplicated from a shared state root, so all contracts, balances, and addresses persist on both children.

What happens on fork: Final-round dispute triggers CourtHub.fork(), creating a new CourtHub child with a new forkId.

A new child chain instance boots from the last common checkpoint state root. No per-application redeploy is required; all addresses and state persist on both children.

The L1 bridge vault for each external asset honors withdrawals only from the child chain endorsed by the asset issuer (e.g., Circle).

1.5.6 Contracts

This section specifies the contract split between L1 and the court L2, the L1⟶L2 command channel, the bond state machine, dispute/settlement flows, fork behavior, and security invariants. The court (staking, disputes, appeals, fork lineage) lives on L1; identity bonds live on the court's L2 and are mutated exclusively by authenticated commands originating from L1 dispute outcomes.

Goals

- L1 as root of truth: staking, slashing, disputes, appeals, and fork lineage.
- L2 for UX and forkability: bonds live with applications, can be frozen/slashed/unfrozen by L1 outcomes, and automatically persist across L2 forks.
- Trust-minimized bridging: a forced-inclusion L1⟶L2 command channel guarantees freezes/settlements are delivered within a bounded time.

Components

L1 (source of truth)

- `CourtHub`: staking, dispute lifecycle, appeals, final settlement, and fork tree. Emits authoritative commands to the child chain.
- `ForkRegistry`: records fork lineage per court and per external asset issuer endorsement.
- `AssetVault`(s): custody external assets; honor withdrawals only from the endorsed L2 child per ForkRegistry.
- (Optional) `StakerFeeCollector`: receives bridged fee flows from L2 for distribution to stakers.

L2 (per-court child chain)

- `CourtInbox`: the only contract allowed to forward L1 commands. Verifies messages from the authorized L1 CourtHub via the canonical cross-domain messenger or proof system with forced inclusion.
- `BondManager`: holds identity bonds; exposes a state machine and mutators callable only by CourtInbox.

- `MatchRegistry` and `Apps`: enroll, check policy, and require “active, unfrozen bond” for actions/withdrawals. They only read bond status; they cannot modify it.

L1→L2 Command Channel

- Uses the chain’s canonical messenger or proof-based bridge with a guaranteed “forced inclusion” bound F (measured in L1/L2 blocks) and replay protection.
- `CourtInbox` stores authorized `CourtHub` and rejects messages from others; commands are idempotent, keyed by (disputeId, bondId, commandType).

Bond State Machine (L2)

- States: ACTIVE, EXITING(t_end), FROZEN(cnt, t_end?), SLASHED, WITHDRAWN.
  - `FROZEN` maintains a reference count `cnt` of active disputes on this bond.
  - `EXITING` starts a withdrawal cooldown with deadline `t_end`.
- Invariants:
  - Only `CourtInbox` may call `freeze`, `unfreeze`, `settleSlash`.
  - `finalizeWithdraw` requires state ∈ {ACTIVE, EXITING} and now ≥ t_end and cnt == 0.
  - Commands are idempotent and order-agnostic across retries/reorgs.
- Withdrawal cooldown W must satisfy: $W \geq F + \Delta$, where Δ covers worst-case reorg/latency margins. This ensures any L1 dispute opened before t_end will deliver a freezeBond command to L2 before withdrawal is finalizable.

Dispute and Settlement Flow

- Open dispute (L1):
  1. Disputer calls `CourtHub.openDispute(chainIdL2, appId, bondId, postedB, κ, fees)`.
  2. L1 locks disputer bond/fees and records the case.
  3. L1 enqueues command: `freezeBond(bondId, disputeId)`.
  4. L2 `CourtInbox.execute` proves the message and calls `BondManager.freeze(bondId, disputeId)`; bond becomes `FROZEN(cnt+1)`.
- Bond withdrawal (L2):
  - `startUnbonding(bondId)` sets `EXITING(now+W)` only if `cnt == 0`.
  - `finalizeWithdraw(bondId)` requires now ≥ t_end and cnt == 0.
  - If a freezeBond arrives while EXITING, the state becomes FROZEN(cnt+1) and finalization is blocked.
- Resolution (L1 ⟶ L2):
  - After the final appeal round, `CourtHub.resolve(disputeId, outcome, slashAmount, splits)` emits one of:

        * `unfreezeBond(bondId, disputeId)` if the user wins (or procedural-only without slash).
        * `settleBond(bondId, disputeId, slashAmount, payDisputer, payStakers)` if the disputer wins (or for procedural partial slashes `slashAmount = ρ·B` while keeping the remainder frozen for subsequent rounds).

- L2 `CourtInbox` executes:
    - `BondManager.unfreeze(bondId, disputeId)` $\longrightarrow$ decrement cnt; if cnt==0 and EXITING deadline passed, user may finalize.
    - `BondManager.settleSlash(bondId, disputeId, slashAmount, payDisputer, payStakers)` $\longrightarrow$ transfer `slashAmount` from the bond to the disputer and to a local `FeeEscrow` for stakers; if the entire bond is slashed, set `SLASHED`; otherwise reduce remaining bond and maintain freeze/refcount per additional disputes.
- Fees to L1 stakers:
    - L2 accumulates staker shares in `FeeEscrow`. A periodic L2⟶L1 bridge transfers them to `StakerFeeCollector` for distribution on L1.
- Multiple disputes:
    - Freezes are reference-counted by `disputeId`.
    - Settlements/unfreezes are idempotent per `(disputeId, bondId)`.
    - Partial slashes reduce the bond balance and may leave it `FROZEN` for remaining active disputes.

Forking Behavior

- Final-round corruption risk can trigger `CourtHub.fork()`:
    - Creates a new L1 `CourtHub` with fresh staking token logic via L1 snapshot/claim contracts (exclude captured stake). The old token persists but loses utility.
    - Boots a new L2 child from the last common checkpoint. All contract addresses and state (including bonds) persist on both children.
    - `ForkRegistry` records lineage; external `AssetVault`(s) honor withdrawals only from the issuer-endorsed child.
    - The honest child's CourtInbox genesis config points to `authorizedCourtHub = CourtHub`. The attacker child remains pointed at the old CourtHub. Apps and BondManager addresses remain identical on both children; only the command authorizer changes.

Security Invariants

- Only L1 `CourtHub` can initiate freeze/unfreeze/settle via `CourtInbox`; apps/users cannot bypass it.
- Forced inclusion guarantees any L1-issued freeze/settle reaches L2 within F; $W \geq F + \Delta$ blocks race-to-withdraw.

- Idempotent, ordered-insensitive command processing; ignore duplicates or stale commands safely.
- ACL:
  - `CourtInbox` stores authorized `CourtHub` and messenger address; both updatable only at fork genesis on the child.
  - `BondManager` trusts only `CourtInbox`.
- Accounting:
  - Bond balances and slashes are monotone; cannot underflow.
  - Partial slashes during procedural defaults never reverse; subsequent substantive outcomes only affect remaining bond.
- Identifiers
  - `bondId` can be keccak256(user, appId, scope), where scope supports per-app or per-action uniqueness.
  - `cmdType` is one of `FREEZE`, `UNFREEZE`, `SETTLE`.
  - `Parameters`
    - `F`: worst-case L1⟶L2 inclusion bound.
    - `W`: withdrawal cooldown on L2, set per court class with $W \geq F + \Delta$.
    - `ρ`: procedural default slash fractions per round (as specified in Dispute Economics).

This split keeps the dispute game, staking, and forking authority on L1 while allowing bonds to live where applications do. Freezes and settlements are authoritative L1 decisions enforced on L2 via a forced-inclusion command channel, preserving UX and enabling clean recovery under forks.

1.5.7 Biometrics Design

Courts choose what biometrics they can verify and the hardware/software requirements for verifying them. We believe most courts will use simple biometrics that can be verified with a smartphone for wider accessibility. This allows users to sign up using their phone, and verifiers to avoid needing specialized hardware. Facial biometrics will be a popular choice for this reason. Palm scans may be another feasible option for smartphones. However these biometrics tend to be less specific and more prone to false matches. Applications using these should allow some tolerance such as less than 10 matches to prevent legitimate users from being unable to use the app. For most applications, small scale sybil attacks have little impact on the app, so allowing users to create say 9 extra accounts is worth the tradeoff of tolerating some false positives. These users would still have to post bonds for all the extra accounts. Courts could potentially allow multi-account disputes where all those accounts have to show up for the same dispute and the verifiers check whether any accounts belong to the same person. This could discourage small scale sybils while allowing tolerance for false positives.

Court of Humanity project will provide a reference app with facial biometrics.

1.5.7.1 Biometrics Privacy

Requirements and Threat Model

- Raw biometrics never leave user control in decipherable form; storage is encrypted or secret-shared with deletion/expiry semantics.
- Queries are purpose-limited: match counts are scoped per application and potentially per action.
- Anti-probing: adversaries cannot enumerate accounts or buckets.

Per-App Scoping and Uploads

- Users enroll separately per application scope. Each enrollment produces app-scoped encrypted artifacts (handles) that cannot be re-used by other apps, preventing cross-app linkage by design. App scoping is enforced by fhevm "bridging" of handles into an app domain so the same ciphertext never yields the same handle across apps. Users submit external values with coprocessor attestations, which contracts verify before converting to internal encrypted types.

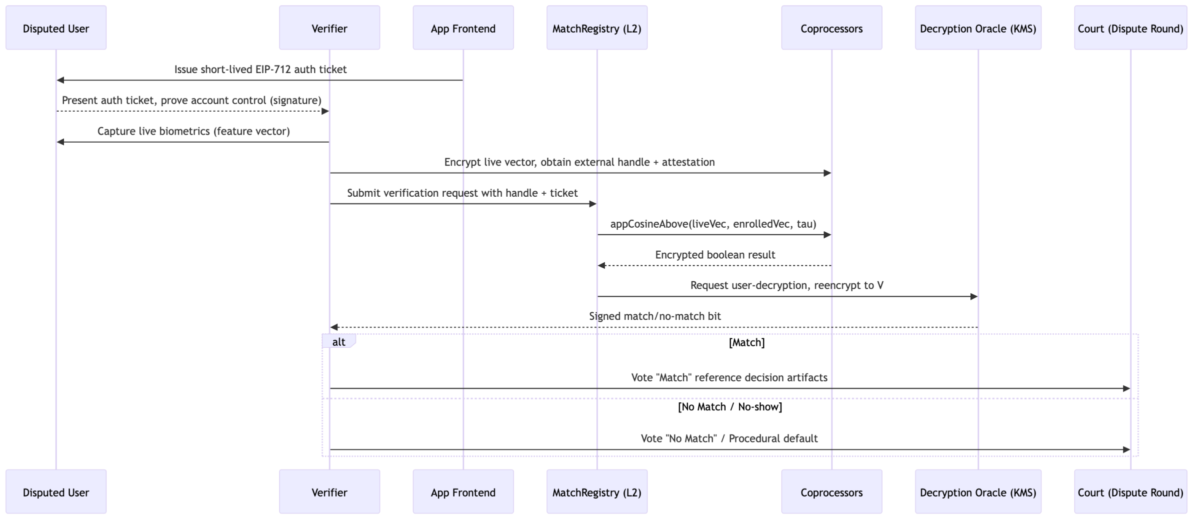
Secure Computation Backend

- fhevm (Zama): contracts operate on encrypted user-defined types (e.g., `euint32`, `ebool`) as opaque handles. Heavy FHE computation (LSH bucket derivation, index updates, candidate retrieval and encrypted similarity scoring) runs off-chain on coprocessors, which commit ciphertext digests to a Gateway. Access is controlled by on-chain ACL; decryptions use a KMS with threshold MPC and signed results. Contracts never see raw biometrics, bucket IDs, or per-bucket contents.

Verifier Protocol (In-Person, Privacy-Preserving)

- Account binding: user proves control of the account (signature) and presents fresh biometrics on the verifier device.
- Limited-use authorization: the user provides a short-lived EIP-712 "auth ticket" authorizing a specific check for a specific purpose/window; results are returned via user decryption (reencrypted to the verifier's public key).
- Outcome: verifiers receive only the minimum necessary signal (match/no-match boolean and integrity checks).

1.5.7.1.1 Verifier–User Verification Sequence


Disputed User
Verifier
App Frontend
MatchRegistry (L2)
Coprocessors
Decryption Oracle (KMS)
Court (Dispute Round)
Issue short-lived EIP-712 auth ticket
Present auth ticket, prove account control (signature)
Capture live biometrics (feature vector)
Encrypt live vector, obtain external handle + attestation
Submit verification request with handle + ticket
appCosineAbove(liveVec, enrolledVec, tau)
Encrypted boolean result
Request user-decryption, reencrypt to V
Signed match/no-match bit
alt
[Match]
Vote "Match" reference decision artifacts
[No Match / No-show]
Vote "No Match" / Procedural default

1.5.7.2 Hierarchical Biometrics

While more accessible biometrics like facial scans are desirable to minimize hardware costs (eg just a smartphone camera) and hardware decentralization, false positive rates are higher. Biometrics can be arranged in a hierarchy where, when two users match on a lower level biometric, they can submit a higher level biometric to determine if they are the same person.

For example, say user A submits a facial scan and posts a bond to perform an action. Later, user B submits a facial scan to perform the same action, but matches user A, and the app policy requires 0 matches. User B can submit a unique iris scan which makes them no longer matching user A, and able to perform the action. User A will be considered to have 1 match until they submit their own unique iris scan. Applications can restrict user A as soon as User B signs up, or they can handle it when user A performs another action.

Courts choose what biometrics they can verify and disputed users will have to match all submitted biometrics.

1.5.7.3 Biometric Drift

Biometric drift is a problem where the biometric data of a user changes over time. This can happen for a variety of reasons, such as a user getting a new haircut, changing their makeup, or even just aging. This can cause the user to no longer match their original biometrics, and can cause disputes to be raised.

Users should be asked by the app to check their own biometrics every time they submit or extend a bond. This "dry run" takes their biometrics and checks that it matches the biometrics on the backend. If not, they can submit updated biometrics. Note that this may change their match count.

1.5.8 fhevm Integration (Architecture and Flows)

This section specifies how biometric data stays encrypted at every stage of the system. Court of Humanity uses fully homomorphic encryption (FHE) so that biometric comparisons — checking whether a new user's face matches anyone already enrolled — happen entirely on encrypted data. The system never sees, stores, or transmits raw biometrics. Instead, smart contracts work with opaque encrypted "handles," and a network of off-chain processors performs the heavy mathematical operations on ciphertexts. The only information that ever emerges is a single yes-or-no answer: does this enrollment collide with an existing one? Even that answer can be kept encrypted when possible. The architecture below details the contracts, operators, and data flows that make this work.

- Components and contracts
  - AppRegistry (public): registers `appId` with an immutable commitment to LSH params; assigns an app-domain for handle derivation.
  - MatchRegistry (confidential): stores per-user unit-normalized fixed-point vector handles and bucket-vector handles; symbolically triggers coprocessor ops; maintains tight ACL (user + registry).
  - BondManager (public): bond deposits/lockups; integrates policy checks; emits disputes; freezes bonds on dispute.
- App-scoped handle domains
  - Handles are "bridged" into an app domain so the same user's encrypted values differ across apps. External-value attestations still bind user `u` and contract `c` and are verified on-chain.
- Precompiled FHE operators (symbolically invoked on-chain, evaluated off-chain)
  - appLshDerive(appDomain, templateHandle, paramsHash) -> bucketVectorHandle
  - appLshReplace(appDomain, oldBv, newBv) // decrement old index entries, increment new
  - appLshCandidates(appDomain, bv, limits) -> candidateSetHandle // private candidate retrieval (multi-probe, capped)
  - appVecDot(fpVecA, fpVecB) -> esint64 // fixed-point dot product
  - appCosineAbove(fpVecA, fpVecB, tauQ) -> ebool // division-free cosine on unit-normalized fixed-point vectors
  - appSimAnyAbove(appDomain, fpVec, tauQ, limits) -> ebool // exists candidate with similarity $\geq \tau$
  - Coprocessors store only ciphertexts and digests; per-bucket contents never appear on-chain.

- Enrollment/update

  1. Frontend computes a unit-length feature vector, quantizes it to fixed-point (e.g., Q1.15 or Q2.14), encrypts the normalized vector and an LSH template via the Gateway, and obtains external handles + attestations.
  2. MatchRegistry verifies attestations, calls appLshDerive, stores the encrypted normalized vector handle, and on first-time calls appLshReplace(ZERO, bv); updates ACL for user + registry.

- Policy enforcement (nearest-neighbor threshold)

  - Compute `exists = appSimAnyAbove(appDomain, fpVec, tauQ, limits)` and `ok = FHE.not(exists)`.
  - No-decrypt gating: use `FHE.select` to nullify side-effects if `ok` is false; tx succeeds, revealing nothing.
  - Minimal-decrypt gating: request decryption of `ok` (one bit) via decryption oracle; `require(ok)`.
  - Optional short-circuit: implementations MAY first check an approximate LSH count and only run fine similarity if the coarse filter indicates potential collisions.

- Controlled authentication (anti-probing)

  - User issues short-lived EIP-712 auth ticket embedding user, allowed contracts, optional verifier(s), expiry, and a reencryption public key.
  - Verifier captures live scan, obtains an external handle, and MatchRegistry computes an encrypted decision using `appCosineAbove(liveVec, enrolledVec, tauQ)`.
  - Use user-decryption; KMS returns encrypted shares reencrypted to the verifier key; only the verifier learns the bit.

- Leakage and unlinkability

  - Contracts learn only handles and optionally one-bit results; no bucket IDs, candidate sets, or similarity scores are exposed.
  - App-domain handles prevent cross-app linkability; per-action context labels can further scope checks.

1.5.9 Division-Free Cosine Similarity (Encrypted)

We avoid division inside FHE by operating on unit-normalized, fixed-point vectors and comparing dot products to a quantized threshold.

- Representation: the client normalizes the template to unit L2 norm and quantizes each component to a fixed-point integer (e.g., Q1.15). This preserves cosine similarity: `cos(x,y) = x·y` when `||x|| = ||y|| = 1`.

- Encrypted dot product: coprocessors evaluate `appVecDot(x,y)` under FHE and obtain an encrypted fixed-point sum in a wide accumulator (e.g., 64-bit). No divisions are performed.
- Threshold test: compare the encrypted dot product to an encrypted/public fixed-point threshold `tauQ` using `appCosineAbove(x,y,tauQ)` to obtain an encrypted boolean.
- Distance equivalence: for unit vectors, squared Euclidean distance satisfies `||x-y||^2 = 2(1 - x·y)`, so a cosine threshold τ is equivalent to a distance threshold without requiring division.
- Performance notes: quantization width and vector dimension determine accumulator size; typical face/iris embeddings (d≈128–512) fit comfortably in 64-bit sums with 15–16 fractional bits. Parameters are app-specific and can be audited.
- Performance and security
- Heavy ops (candidate retrieval and top-K dot products) run on coprocessors; Gateway stores commitments to ciphertext digests for public verifiability. LSH reduces work to a small candidate set (configurable limits) before fine similarity.
- KMS is threshold-secured (t < n/3); decryption/oracle results are signed and verifiable on-chain.

1.6 Bonafide: Social Identity Layer

1.6.1 Design Philosophy

Bonafide is a minimal social identity application launched and maintained by the Court of Humanity team as a foundational component of the ecosystem. Its purpose is not to replicate the feature sets of existing social platforms, but to provide the minimum viable reputation surface for sybil detection: profiles, posts, reposts, and follows. By keeping the feature set barebones, Bonafide avoids competing with richer decentralized social platforms like Farcaster or Lens, which may integrate Court of Humanity bonds and disputes to serve as alternative or complementary social layers.

Bonafide exists because the dispute mechanism's effectiveness is directly proportional to the observable context available about each account. Without a venue where accounts build persistent, observable histories, disputers lack the confidence to risk their dispute bonds, and the detection probability D remains too low for practical bond sizing.

1.6.2 Persistent Identity Bonds

Unlike per-action bonds in downstream applications, Bonafide requires a persistent identity bond to maintain an active account. This bond serves as the user's base-layer identity stake:

- Users post a bond (e.g., $5) to create a Bonafide account and begin posting.
- The bond remains active as long as the user maintains their account.

- To withdraw, users initiate a cooldown period (e.g., 2 weeks) during which the account cannot post but the bond remains disputable.
- If a dispute is raised during cooldown, the bond is frozen and the withdrawal is blocked until resolution.

This persistent bond creates a standing economic exposure that incentivizes legitimate behavior and gives disputers a reliable target. Downstream applications require additional per-action bonds on top of this base-layer stake.

1.6.3 Detection Surface

Bonafide provides disputers with rich, naturally occurring signals for sybil detection:

- **Profile authenticity**: AI-generated profile pictures, stock photos, or missing biographical details.
- **Content patterns**: Formulaic posts, repetitive messaging, propaganda-like content, or suspiciously low engagement.
- **Temporal signals**: Account age, posting frequency, activity patterns consistent with bot behavior (e.g., posting at regular intervals around the clock).
- **Social graph**: Clusters of accounts that follow each other but have little external engagement; unnatural follower/following ratios.
- **Cross-signal correlation**: Groups of accounts created around the same time, posting similar content, with similar profile structures.

These signals allow human observers to form high-confidence beliefs about account authenticity — precisely the kind of confidence needed to risk a dispute bond.

1.6.4 Multi-Jurisdiction Deployment

Bonafide deploys on every court L2 where Court of Humanity instances operate. The Court of Humanity team will launch a starter set of regional court instances (detailed in the Starter Court Rollout section of the Roadmap) alongside a toolkit for others to launch courts in additional jurisdictions.

Users bond into their local jurisdiction. Frontends and indexers aggregate content and profiles across all court L2 deployments, presenting a unified social experience while maintaining the jurisdictional isolation required for court forking.

1.6.5 Attestations for Downstream Applications

The key mechanism connecting Bonafide to the rest of the ecosystem is zero-knowledge attestable credentials. Bonafide issues attestations about account standing that downstream applications can require as prerequisites for bonding or performing actions. Examples:

- "I have an active Bonafide account with an undisputed bond, created at least 3 months ago."
- "I have an active Bonafide account with 50+ posts and no history of failed disputes."

These attestations are unlinkable: the downstream application learns that the user meets the criteria but not which Bonafide account they control. This is accomplished through zero-knowledge proofs over Bonafide's on-chain state.

Downstream applications benefit in two ways:

1. **Inherited detection probability**: Sybil accounts must first survive scrutiny on Bonafide, where context is richest and detection probability is highest. Only accounts that pass this filter can generate valid attestations.
2. **Reduced bond requirements**: Because Bonafide's high D acts as a pre-filter, downstream apps can set lower per-action bond sizes than they would need in isolation, improving usability.

Applications can also accept direct Bonafide bond linkage (foregoing unlinkability) for even stronger guarantees, offering users a privacy-cost tradeoff.

1.6.6 Privacy Architecture

Bonafide's centrality raises legitimate privacy concerns. The system addresses these through several mechanisms:

Cross-application unlinkability. Bonafide attestations use zero-knowledge proofs so downstream applications cannot determine which Bonafide account a user controls. Combined with the existing per-app biometric handle scoping, this prevents linking a user's social identity to their activities in other applications.

Tiered privacy. Applications can offer a privacy spectrum:

- *Linked accounts*: Users who voluntarily connect their Bonafide profile to a downstream app identity receive lower bond requirements, since disputers have maximum context (highest D).
- *Attested accounts*: Users prove Bonafide standing via ZK attestation without revealing their identity. Moderate bond requirements.
- *Unattested accounts*: Users with no Bonafide linkage at all. Highest bond requirements, as detection relies solely on in-app context.

Dispute firewalling. A dispute raised against a user's downstream app identity proceeds through its own court process without revealing the user's Bonafide identity. Verifiers in the downstream dispute check biometrics against the downstream app's enrolled data, not Bonafide's. The two identities remain separated unless the user has opted into linkage.

Batched proceedings. Courts can process disputes in batches to reduce temporal correlation attacks where an observer might link a Bonafide account to a downstream account by matching dispute timing.

1.6.7 Integration Path for Existing Social Platforms

Bonafide is intentionally minimal because the long-term social layer for Court of Humanity may be an existing decentralized social platform. If platforms like Farcaster or Lens integrate Court of Humanity bonds and disputes:

- Users on those platforms could post persistent identity bonds and be subject to disputes, just as on Bonafide.
- Those platforms could issue attestations equivalent to Bonafide's, accepted by downstream applications.
- Bonafide would remain available as a credibly neutral, protocol-native fallback that does not depend on any third-party platform's governance or continued operation.

The attestation interface is designed to be platform-agnostic: downstream applications specify attestation criteria (account age, activity thresholds, dispute history) and accept attestations from any qualifying social layer, whether Bonafide or an integrated third-party platform.

1.7 Application Integration

Applications integrate with Court of Humanity through biometric policy enforcement, dispute hooks, and — for most applications — Bonafide attestation requirements. Requiring Bonafide attestations allows applications to inherit the high detection probability of the social identity layer, reducing the bond sizes needed to deter sybil attacks.

1.7.1 Bonafide Attestation Requirements

Applications can require Bonafide attestations as a prerequisite for bonding or performing actions. Attestation criteria are configurable per application:

- Minimum account age (e.g., 3 months)
- Minimum activity level (e.g., 50+ posts)
- Clean dispute history (e.g., no failed disputes)
- Active persistent bond

Because attestations are zero-knowledge, the application learns only that the user meets the criteria, not which Bonafide account they control. Applications that accept attestations from multiple social layers (Bonafide plus integrated third-party platforms) specify a common criteria interface.

The bond sizing heuristic can account for Bonafide requirements by using a higher D estimate when attestations are required, resulting in lower per-action bond sizes.

1.7.2 Scoping and Interfaces

- Match scoping: apps query for existence of similar neighbors within their enrolled users, or narrower subsets (e.g., “voters on proposal X”) using context labels, via LSH candidates plus encrypted cosine checks. Typical policy: require no candidate above threshold τ.
- Enforcement: either no-decrypt gating (nullify side-effects when policy fails) or minimal-decrypt gating (decrypt one boolean via oracle and `require(ok)`).
- Reenrollment: if matches change (e.g., user updates biometrics), apps can recheck or revoke based on policies. Users with active bonds should avoid re-enrollment to ensure they can pass in-person verification if disputed.

1.7.3 Dispute-Dependent Actions (DDA)

For high-stakes operations, applications can declare actions “pending” until all disputes of involved accounts are resolved.

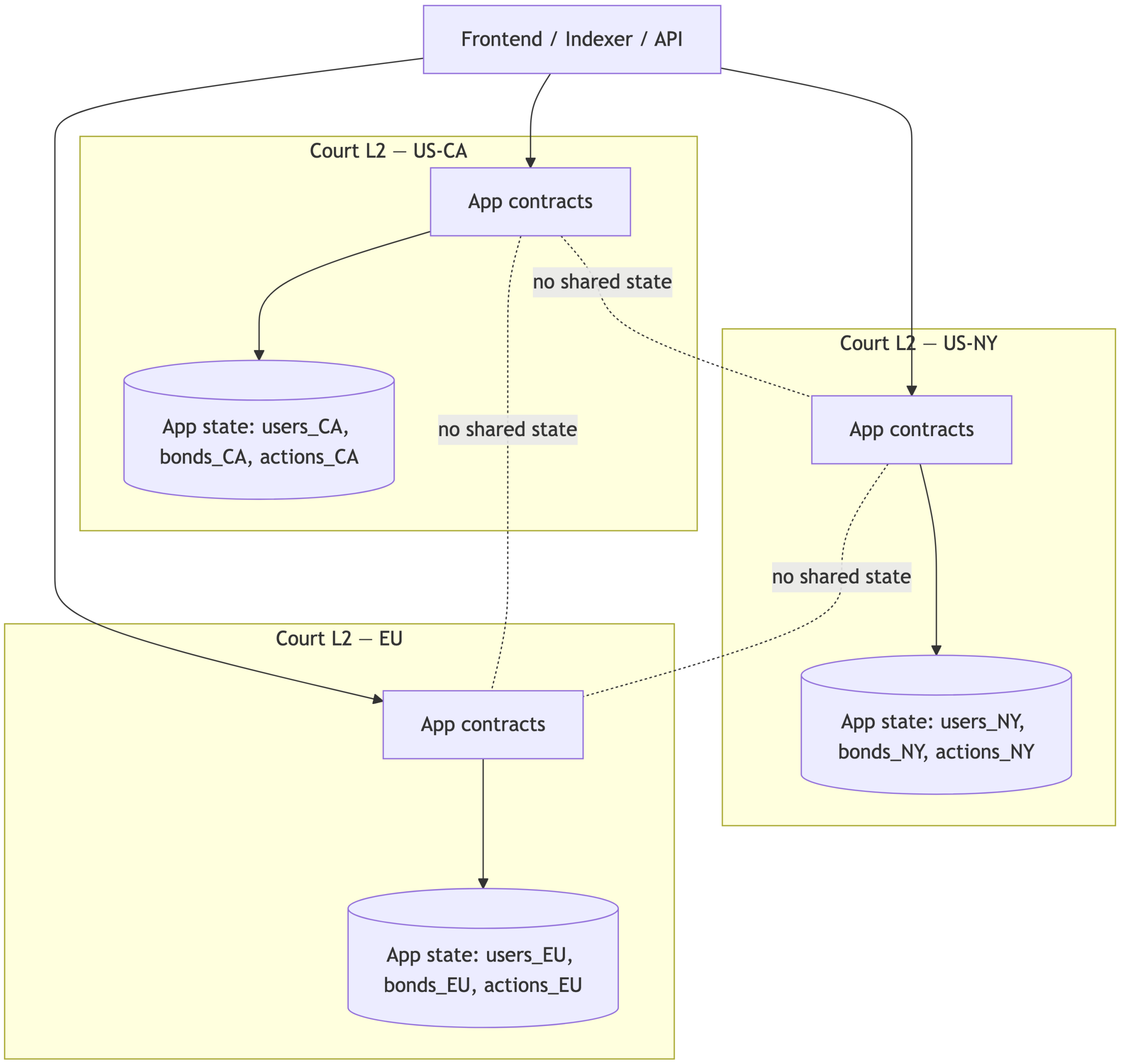
- Example: a DAO fund transfer executes only after a post-vote dispute window; blocs of suspicious voters can be disputed, shifting the outcome if forgeries are confirmed.
- Policy knobs: dispute windows by action type; required incremental bonds for actions attracting scrutiny.
- Usability: reserve DDA for high-value operations; keep routine actions non-DDA to preserve user experience.

1.7.4 Multi-jurisdiction Applications

Applications should deploy on every court subchain they want to accept bonds in. This fragments state and requires multiple L2 node processes but allows forking state along with courts. If applications deploy on the L1, they would need a programmatic fork choice rule to determine which court to use in a fork, or be vulnerable to 51% attacks. We may see some such applications use a majority signal from asset issuers who have to choose what fork to honor assets in.

When summing bonds across courts, applications can use weightings by staked marketcap or other metrics to account for varying security levels.

The diagram below shows how an application’s state is intentionally fragmented across multiple court L2s, with each deployment maintaining isolated state. Frontends/indexers aggregate views by reading from each court chain separately.



1.8 Security Analysis

51% Attacks and Bribery

- Attackers profit only if the value gained exceeds the effective cost to capture the court (stake purchase or bribe of majority-by-stake across appeal rounds).

fhevm Trust and Verifiability

- ACL: only the user, the registry, and explicitly allowed contracts can access encrypted handles.
- Coprocessor commitments: ciphertext digests are committed to the Gateway; anyone can recompute and verify, mitigating substitution/selective-failure risks.
- Threshold KMS: decryption correctness and secrecy rely on an honest minority (e.g., $t < n/3$); results are signed by parties and verifiable on-chain or client-side for user decryption.
- Oracles and relayers are untrusted: signatures bind results; at worst they can be liveness bottlenecks, not correctness faults.

Two-Stage Matching Robustness

- LSH is used only for private candidate retrieval; final match decisions rely on encrypted cosine similarity against normalized vectors. This reduces false positives relative to LSH-only gating while preserving privacy (only a one-bit result is revealed to contracts).

Griefing Resistance

- Dispute fees and loser-pays mechanics deter frivolous disputes; coherent voters are rewarded from incoherent voters.
- Assurance bonds prevent users from griefing verifiers by not showing up for disputes.

1.8.1 Bought Sybils

A natural concern is that an adversary could bypass biometric forgery entirely by recruiting real people to enroll genuine biometrics, post bonds, and then hand over account control. This “bought sybil” attack sidesteps anti-spoofing defenses because the enrolled biometrics are authentic. However, the dispute mechanism creates structural deterrents that make account selling inherently unstable.

Seller recapture. After selling an account, the seller retains the biometrics enrolled to it. At any point, the seller can dispute the sold account. The buyer cannot pass in-person verification — the enrolled biometrics are the seller’s, not the buyer’s — so the dispute succeeds and the bond is slashed to the seller. The seller thus holds a perpetual option to reclaim the bond on top of whatever they were paid for the account. No rational buyer purchases an account when the seller can unilaterally betray at any time, and no enforceable contract can prevent it (one cannot seek legal remedy for a sybil purchase agreement). This produces a lemon market: account selling is a game where the seller’s dominant strategy is to defect, so the market cannot clear.

Identity burning. Even if the seller credibly commits to never disputing (e.g., through reputation in a criminal network), selling an account permanently degrades the seller’s own identity in the ecosystem. If the seller later enrolls in any Court of Humanity–integrated application, their biometrics will match the sold account. Both accounts are then flagged as sybils for any application with a uniqueness requirement, blocking the seller from participation. The seller has effectively burned their biometric identity across all current and future integrated applications.

Residual attack surface. The combination of recapture risk and identity burning narrows the viable attack to recruiting individuals who (a) have no intention of ever using the ecosystem and (b) will credibly forgo the dispute recapture option. This is a significantly more expensive and fragile operation than fabricating identities: the adversary must pay a premium sufficient to compensate for permanent identity loss and the forgone recapture payoff, and must trust that recruits will not defect. The per-identity cost scales with bond sizes and ecosystem utility, making large-scale bought-sybil attacks progressively less economical as the system grows.

1.9 Example Applications

The following applications illustrate how different use cases build on Court of Humanity and Bonafide. Most require Bonafide attestations as a prerequisite for participation, inheriting the social layer's high detection probability while adding context-specific bonds for their own actions.

1.9.1 UBI Prediction Market

Users start off with a certain amount of points they can use to bet on prediction markets. New points are awarded each week as a sort of UBI, to replenish zeroed out accounts. The "total return" of an account can be used as an indicator of predictive skill. Markets with real cash bounties can be created that pay out to the top performers. Users can display their prediction accuracy on Bonafide, adding trust to high accuracy accounts. A Bonafide attestation is required to claim weekly UBI points, preventing sybils from farming multiple allocations.

1.9.2 Democracy Launchpad

Democracy Launchpad is a platform to launch democratic DAOs on court subchains. Membership can be open to anyone willing to bond in that jurisdiction or require more specific criteria such as attending local meetups. Court of Humanity's sybil-resistance allows loose criteria for membership while minimizing the risk of sybil attacks passing malicious proposals. This can allow non-profit organizations to manage funds in a democratic way. Proposals are dispute-dependent actions, which protects against sybil attacks.

When a contentious vote is close, observers can examine the Bonafide profiles of accounts on the winning side. If a cluster of thin accounts with similar posting patterns all voted the same way, those accounts can be disputed before the DDA window closes and the proposal executes. Requiring Bonafide attestations (e.g., 6-month account age, active posting history) raises the cost of DAO manipulation significantly, since attackers must maintain convincing social personas over extended periods.

1.9.2.1 Federation of Democracy DAOs (DAO-of-DAOs)

Because democracy DAO treasuries must reside on court L2s — and only users bonded into that court can vote — democratic governance is inherently local. A user in Houston cannot meaningfully participate in a Northeast court DAO because they would need to travel to Philadelphia or New York to defend their bond if disputed. Selecting a single large court as the “global” home for democracy is therefore impractical: it excludes anyone unwilling or unable to travel to that court’s jurisdiction for disputes.

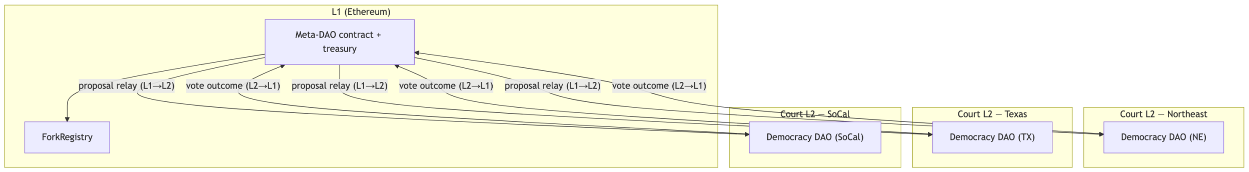
A *federation* of democracy DAOs addresses this by letting each local democracy DAO vote internally on cross-jurisdictional proposals, then relay its outcome to a meta-level DAO-of-DAOs that aggregates the results. Each local DAO counts as one weighted vote in the federation. This preserves the local bond/dispute guarantees of each court while enabling coordinated global decision-making.

1.9.2.1.1 Architecture

The meta-DAO contract lives on L1 (Ethereum), the same layer where CourtHub, ForkRegistry, and AssetVaults already reside. This is the natural choice because:

- L1 is jurisdictionally neutral — not subject to any single court’s fork dynamics.
- The ForkRegistry is on L1, giving the meta-DAO direct access to fork lineage data for each member court.
- The meta-DAO treasury holds assets on L1 directly (or in its own vault), avoiding the question of which court L2 to trust with shared funds.

Each member democracy DAO lives on its court L2, where it conducts local votes with full DDA protections. The existing L1↔L2 messaging channels (the same ones used for dispute commands and fee bridging) relay proposals downward and vote outcomes upward.



1.9.2.1.2 Vote Relay Protocol

1. **Proposal submission.** A proposal is submitted to the meta-DAO on L1, specifying: the action to execute, a voting deadline, and which member courts are eligible.
2. **Downward relay.** The meta-DAO broadcasts the proposal to each eligible member DAO via the L1→L2 command channel (the same forced-inclusion channel used for dispute commands).

3. **Local voting.** Each member DAO conducts a standard local vote. Only accounts bonded into that court and meeting the DAO's Bonafide attestation requirements may vote. The vote is a DDA: a dispute window follows the voting period during which suspicious voters can be challenged.
4. **Outcome relay.** After the local DDA window closes (all disputes resolved or expired), the member DAO relays its outcome — pass, fail, or abstain — along with its current membership count and total active bonds, to the meta-DAO on L1 via L2→L1 messaging.
5. **Meta-aggregation.** The meta-DAO aggregates outcomes according to its weighting rule once a quorum of member DAOs have reported or the deadline has passed.
6. **Meta-DDA window.** Before execution, the meta-DAO opens its own dispute window during which member DAOs or external observers can flag issues: a court fork occurred during the vote, a local outcome was relayed from a now-unrecognized fork, or a member court's local vote was visibly corrupted.
7. **Execution.** If no issues are raised (or raised issues are resolved), the meta-proposal executes on L1 — transferring funds, updating parameters, or taking whatever action was specified.

1.9.2.1.3 Weighting Schemes

How to weight each member DAO's vote is a governance design choice. Several options are available, each with distinct tradeoffs:

One-DAO-one-vote (federalism). Each court gets one vote regardless of size. Simple and egalitarian among jurisdictions, but gives small courts disproportionate per-capita power. Vulnerable to court proliferation attacks: an adversary could spin up many small courts to accumulate meta-votes. Requires strict membership admission criteria to mitigate.

Bonded-member-count weighted (proportional representation). Each court's vote is weighted by its number of bonded members. The most directly democratic: one person, one (indirect) vote. Since bonding requires unique biometrics, this is a reliable proxy for constituency size. Gives large courts correspondingly large influence, which may concern small courts.

Degressive proportionality (square-root weighting). Weight each court's vote by the square root of its bonded member count. This is analogous to allocation rules used in the EU Parliament: smaller courts get more weight per capita than large courts, but large courts still have more total weight. A useful compromise that encourages participation from small courts without allowing them to be dominated.

Total-active-bonds weighted. Weight by the aggregate bond value posted in each court. This captures both constituency size and depth of commitment, but introduces a plutocratic lean: wealthier jurisdictions get more say, and users who post larger bonds (for higher-risk applications) implicitly carry more weight in meta-governance.

We recommend **bonded-member-count weighted** as the default for organizations whose goal is democratic governance, with **degressive proportionality** as a reasonable alternative when the federation includes courts of highly unequal size. The meta-DAO's governance itself can vote to change the weighting rule via a meta-proposal.

1.9.2.1.4 Meta-DAO Treasury

The meta-DAO treasury lives on L1. Member DAOs can contribute funds via L2→L1 transfers. Because L1 is not subject to any court's fork, the treasury is jurisdictionally neutral. For external assets (e.g., USDC), the meta-DAO holds them in a standard L1 contract — no asset vault fork-choice logic is needed because the treasury is not on a court L2.

An alternative model avoids a central treasury entirely: meta-votes produce *recommendations* that each local DAO independently chooses whether to fund from its own treasury. This is weaker (non-binding) but eliminates the single-point-of-failure risk of a pooled L1 treasury. A hybrid is also possible: an L1 treasury for shared cross-jurisdictional projects, with local treasuries for jurisdiction-scoped spending.

1.9.2.1.5 Fork Handling

Court forks are the most complex challenge for a multi-jurisdiction federation. When a member court forks, the meta-DAO must determine which fork to recognize and how to handle any in-flight votes.

Fork detection. The meta-DAO monitors the ForkRegistry on L1. When a `CourtHub.fork()` is registered for a member court, the meta-DAO automatically flags that court's membership as *suspended*.

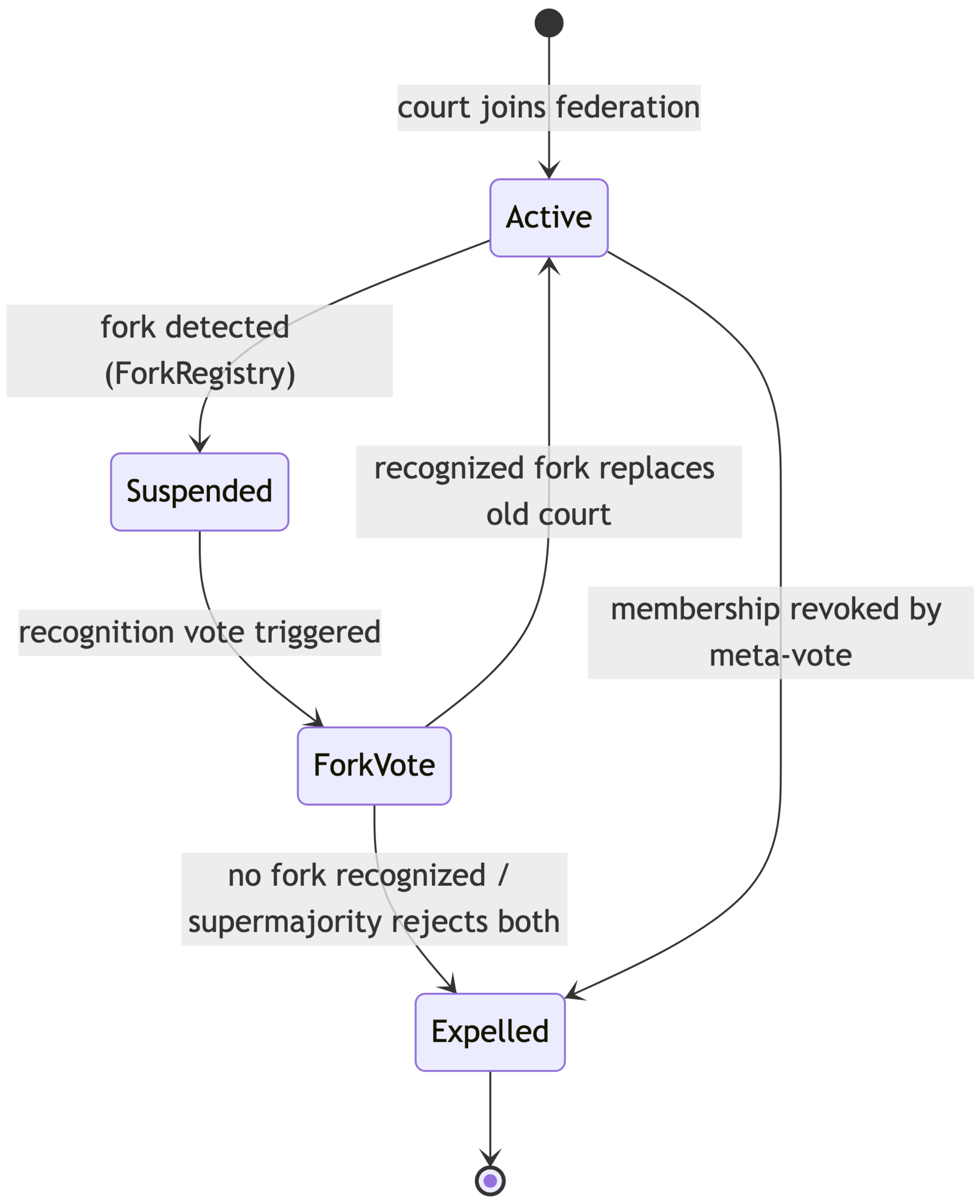
In-flight vote handling. Any pending meta-proposal that has received (or is awaiting) a vote from the forking court enters a *paused* state. The pause persists until the fork is resolved. Other member courts' votes remain valid; only the forking court's contribution is held.

Fork recognition vote. The non-forking member courts vote on which fork to recognize. This mirrors how asset issuers choose forks via the ForkRegistry, but the decision is made democratically by the federation rather than by a single issuer. The recognized fork replaces the old court in the meta-DAO's membership registry. A supermajority (e.g., 2/3 of non-forking courts by weight) should be required to prevent hasty or contested recognition.

Vote resubmission. Once a fork is recognized:

- If the fork affected the local vote outcome (e.g., sybil voters on the winning side were excluded in the honest fork), the recognized fork's DAO re-conducts its local vote and resubmits to the meta-DAO.
- If the fork did not affect the local vote (e.g., the forking dispute was unrelated to the meta-proposal), the recognized fork's DAO can ratify the previous outcome without re-voting.

Cascading forks. If multiple courts fork simultaneously (a coordinated attack or coincidence), each fork is handled independently. The meta-DAO pauses all affected votes and resolves forks one at a time or in parallel, depending on governance rules. A minimum quorum of non-suspended courts is required to conduct fork recognition votes — if too many courts are forking simultaneously, the meta-DAO halts until sufficient courts stabilize.


court joins federation
Active
fork detected (ForkRegistry)
Suspended
recognized fork replaces old court
recognition vote triggered
membership revoked by meta-vote
ForkVote
no fork recognized / supermajority rejects both
Expelled

1.9.2.1.6 Membership Governance

To prevent court proliferation attacks and ensure quality, the meta-DAO governs its own membership:

- **Admission criteria.** New courts must meet minimum thresholds: minimum bonded member count, minimum total active bonds, minimum court age, and endorsement by existing member courts. Admission is itself a meta-proposal subject to vote.
- **Expulsion.** A member court can be expelled by supermajority meta-vote if it is found to be compromised, inactive, or operating in bad faith.
- **Quorum.** Meta-proposals require a minimum quorum (e.g., 50% of member courts by weight) to be valid, preventing a small subset from acting unilaterally.
- **Supermajority for critical actions.** Treasury transfers above a threshold, membership changes, and weighting rule changes require a supermajority (e.g., 2/3 by weight).

1.9.2.1.7 Safeguards Against Meta-Capture

The meta-DAO has no staking token of its own and therefore cannot be forked in the same way as a court. Its security derives from the federation structure: corrupting the meta-DAO requires simultaneously corrupting a majority (by weight) of member courts, each of which has its own independent staking token, verifier pool, and fork mechanism. Additional safeguards:

- **Futarchy for critical decisions.** The meta-DAO can use futarchy (conditional prediction markets) for its most consequential proposals, the same mechanism recommended for court DAOs.
- **Timelock and veto.** Passed meta-proposals enter a timelock during which any member court can veto execution by triggering a re-vote with a higher supermajority threshold.
- **Rate limiting.** Cap the total treasury outflow per epoch to limit damage from a compromised vote.

1.9.2.1.8 Limitations

Latency. The full lifecycle of a meta-vote is long: proposal relay (hours), local voting period (days to weeks), local DDA window (days), L2→L1 relay (hours to days depending on the bridge), meta-aggregation, and meta-DDA window (days). Total time could be 3–6 weeks. This makes the federation suitable for major governance decisions but not day-to-day operations, which should remain local.

Complexity. The federation adds a layer of governance on top of already-complex court and DAO mechanics. Each meta-vote involves cross-chain messaging across potentially dozens of court L2s, fork monitoring, and multi-phase dispute windows. Implementation and operational overhead are significant.

Incomplete fork isolation. If the meta-DAO treasury on L1 disburses funds to a court L2 that subsequently forks, those funds are now subject to that court's fork dynamics. The meta-DAO can mitigate this by disbursing to L1 addresses or by requiring recipient courts to hold funds in asset-vault-like structures with meta-DAO fork-choice authority, but this adds yet another layer of complexity.

No meta-fork mechanism. If a supermajority of member courts collude to capture the meta-DAO, there is no fork escape hatch at the meta level. The defense is that such collusion requires independently capturing multiple courts, each with its own economic security. If this ever occurs, the honest minority courts can simply leave the federation and form a new one.

1.9.3 Sybil-Resistant Airdrops & Retroactive Rewards

- **Purpose**: Fair token/user rewards without bot swarms.
- **Mechanics**: Require a Bonafide attestation plus a per-claim identity bond; payouts are DDA until the dispute window closes; suspicious clusters can be identified by examining claimants' Bonafide profiles and disputed.
- **Policy knobs**: Bond sized to grant size; attestation criteria (e.g., minimum account age); per-campaign scope; short dispute windows for UX.

1.9.4 Quadratic Funding & Public Goods Grants

- **Purpose**: Prevent fake donors inflating match amounts.
- **Mechanics**: Donors bond per proposal; matching is DDA post-vote; disputes on correlated donors trigger in-person checks.
- **Policy knobs**: Higher κ for governance-class risk; per-proposal context labels.

1.9.5 Anti-Scalping Ticketing & Event Access

- **Purpose**: Enforce 1-person-1-ticket and deter bot resale.
- **Mechanics**: Bond to reserve/buy; transfers require bonded recipient; optional venue-side verifier checks for high-value tiers.
- **Policy knobs**: Graduated bonds by ticket tier; short local-court dispute windows.

1.9.6 Referral, Bounty & Ambassador Programs

- **Purpose**: Stop referral farming and sockpuppet bounties.
- **Mechanics**: Referrer and referee keep small bonds until retention criteria; suspected rings are disputable; slashes redistribute.
- **Policy knobs**: Bond scales with reward; rolling DDA release after milestones.

1.9.7 Reviews & Ratings Integrity (App Stores, Marketplaces)

- **Purpose**: Thwart paid/fake review floods.
- **Mechanics**: Bonded reviews; batch disputes on correlated submissions; slashes fund auditors and stakers.
- **Policy knobs**: Tiny bonds for UX; auto-raise disputes on anomaly scores.

1.9.8 Proof-of-Personhood Messaging & Email

- **Purpose**: Spam-resistant communication with privacy.
- **Mechanics**: Senders maintain a small rolling bond; recipients can filter to active-bond senders; spam flags can trigger disputes.
- **Policy knobs**: Per-domain scope; grace periods for new accounts; bonded rate limits.

1.9.9 Limit-1 Promotions & Free Trials

- **Purpose**: Stop coupon and promo abuse.
- **Mechanics**: App-scoped uniqueness check per campaign; small bond per redemption; disputes on clusters or reuse.
- **Policy knobs**: τ and candidate limits tuned for high scale; brief windows.

1.9.10 Dating & Community Admission

- **Purpose**: Human-only spaces; reduce catfishing and bot spam.
- **Mechanics**: Bond to DM or join; reports can open disputes; rare in-person checks by local verifiers.
- **Policy knobs**: Low social-class bonds; rate-limited disputes to reduce harassment.

1.9.11 P2P Lending & Microcredit

- **Purpose**: Prevent loan stacking via sybils.
- **Mechanics**: Borrowers post identity bonds tied to loan tiers; defaults or fraud disputes can slash; repeat offenders face stricter τ and limits.
- **Policy knobs**: Bond scales with exposure; lender-pooled disputes.

1.10 Related Work

- Decentralized arbitration: Kleros demonstrates Schelling-based juries, coherent/incoherent stakes, and appeals; we adapt this to identity disputes and align fees to staker security needs.
- Proof of Unique Human (PoUH): prior approaches range from web-of-trust ceremonies to hardware iris devices. Hardware-trust solutions concentrate power in device key registries and are susceptible to bribery/blackmail; web ceremonies have locality and scalability limits.
- Privacy-preserving similarity search: MPC and FHE lines of work, LSH for encrypted candidate retrieval, and encrypted index structures inform our private uniqueness design.

1.11 Roadmap

- Phase 1 (MVP): Launch Bonafide alongside starter court instances (see Starter Court Rollout below). fhevm-backed uniqueness (encrypted enrollment, LSH-based candidate retrieval, division-free cosine checks on encrypted vectors, consented verifier checks), basic dispute courts, persistent bond posting, verifier mobile tooling. Court launcher toolkit for community-operated jurisdictions.
- Phase 2: Bonafide ZK attestation infrastructure for unlinkable cross-app reputation. DDA workflows for high-stakes actions. SDKs for existing social platforms (Farcaster, Lens) to integrate Court of Humanity bonds and disputes. Richer governance and analytics; performance tuning (bucket packing, batching, SIMD dot products) on coprocessors.
- Phase 3: Multi-platform attestation aggregation and tiered privacy bonding. Verifiable FHE/ZK attestations for coprocessor computations, hardware acceleration, scalable encrypted indices (multi-probe LSH, hierarchical modalities), cross-chain court bridging.

1.11.1 Starter Court Rollout (Continental US)

The Court of Humanity team will organize the launch of starter courts. These initial courts are designed to balance economic security — concentrating enough users, bonds, and verifiers per court instance to make stake pools meaningful — with practical travel constraints. Subcourts handle initial dispute rounds locally; only appeals escalate to the top-level hub. Hub cities are chosen for geographic centrality so that worst-case drive time from any subcourt stays under roughly four hours, with occasional exceptions for high-population metros whose inclusion materially strengthens the court.

Courts are rolled out in waves, prioritizing regions with the largest combined metro populations (and thus the strongest bootstrapping potential for stake and verifier pools) first.

1.11.1.1 Wave 1 — Anchor Courts

These courts cover the highest-population corridors and are expected to attract the deepest stake pools and most active verifier communities at launch.

Court of the Northeast (~40M metro pop) — Hub: Philadelphia. Subcourts: New York City, Washington DC, Baltimore, Boston. Philadelphia sits at the geographic center of the corridor: ~2 hr to NYC, ~2.5 hr to DC, ~1.5 hr to Baltimore. Boston is the outlier at ~5 hr but is too large to exclude; initial disputes resolve locally and only rare appeals require the trip to Philadelphia.

Court of Texas (~20M metro pop) — Hub: Austin. Subcourts: Dallas–Fort Worth, Houston, San Antonio. Austin is centrally positioned with every subcourt within 3 hr. Texas has a large and active crypto community, making verifier recruitment straightforward.

Court of Southern California (~18.5M metro pop) — Hub: Los Angeles. Subcourts: San Diego, Las Vegas. San Diego is a 2 hr drive; Las Vegas at ~4.5 hr is borderline but has no closer natural court. If the travel constraint proves too strict, Las Vegas can split into a standalone single-city court.

Court of Northern California (~10M metro pop) — Hub: San Francisco. Subcourt: Sacramento (~1.5 hr). The Bay Area's tech concentration should drive strong per-capita verifier participation and stake.

1.11.1.2 Wave 2 — Regional Courts

Launched after Wave 1 courts demonstrate viable dispute flow and verifier participation, these courts extend coverage to the Midwest, Ohio Valley, and Southeast.

Court of Florida (~13.5M metro pop) — Hub: Orlando. Subcourts: Miami, Tampa, Jacksonville. Orlando is the geographic center of Florida's major metros, with every subcourt within 3.5 hr.

Court of the Great Lakes (~13M metro pop) — Hub: Chicago. Subcourts: Milwaukee, Indianapolis. Chicago's large metro area anchors the court; Milwaukee is 1.5 hr away and Indianapolis 3 hr.

Court of the Ohio Valley (~13M metro pop) — Hub: Columbus. Subcourts: Detroit, Cleveland, Cincinnati, Pittsburgh. Columbus is centrally located with every subcourt within 3.5 hr. This cluster of mid-size metros aggregates into a substantial combined population.

Court of the Southeast (~11M metro pop) — Hub: Atlanta. Subcourts: Charlotte, Nashville. Both subcourts are ~3.5 hr from Atlanta. Birmingham and Memphis are candidates for later subcourt additions.

1.11.1.3 Wave 3 — Expansion Courts

Smaller or more geographically isolated courts that round out continental coverage once the network effect from earlier waves reduces bootstrapping risk.

Court of the Pacific Northwest (~6.5M metro pop) — Hub: Seattle. Subcourt: Portland (~3 hr).

Court of Phoenix (~6M metro pop) — Hub: Phoenix. Optional subcourt: Tucson (~2 hr). Phoenix is the fifth-largest US metro but geographically isolated from other major cities.

Court of Minneapolis (~3.7M metro pop) — Single-city court. Minneapolis–St. Paul is ~6 hr from Chicago, too far for any multi-city grouping.

Court of Denver (~3M metro pop) — Single-city court. Denver is 6+ hr from the nearest major metro in any direction. A strong local tech community supports a viable verifier pool despite the smaller population.

Court of Salt Lake City (~1.3M metro pop) — Single-city court. Salt Lake City is geographically isolated (~6 hr to Denver, ~5 hr to Boise) but has an outsized crypto community relative to its metro size, making it a strong candidate for early verifier recruitment.

1.11.1.4 Coverage and Expansion Path

At full rollout the thirteen starter courts cover approximately 156–161M people in direct metro populations, or roughly 48% of the continental US. Including surrounding areas within driving distance, effective coverage reaches an estimated 55–60%.

Notable uncovered metros — St. Louis, Kansas City, New Orleans, Oklahoma City — are candidates for future courts or subcourt additions as the network grows. The court launcher toolkit shipped in Phase 1 enables community-operated courts in these and international regions without waiting for official rollout waves.

1.12 Go-to-Market Strategy

1.12.1 The Adoption Challenge

Court of Humanity asks more of its users than existing identity systems: posting bonds, maintaining social profiles on Bonafide, and potentially appearing in person for disputes. This friction is the direct cost of removing trusted intermediaries and hardware trust assumptions. The system cannot succeed by minimizing this friction below what the mechanism requires; instead, it must reach users at the point where the cost of *not* having sybil resistance exceeds the cost of participation.

This is not a hypothetical threshold. Sybil attacks are an escalating crisis across digital systems. Social media platforms are increasingly overrun by bot accounts that spread propaganda and crowd out human interaction. Generative AI has made fake media — video, audio, images — indistinguishable from authentic content, enabling deepfake scams and large-scale disinformation campaigns. Online polls and governance votes are trivially manipulable. Reputation systems on marketplaces are flooded with fabricated reviews. As these problems intensify, they will erode public trust in institutions, online discourse, and democratic processes. The friction of bonding and verification will look increasingly modest against the alternative: a digital environment where nothing can be trusted and no collective decision can be taken at face value.

The strategy is therefore not to convince users that friction is acceptable in the abstract, but to build and deploy applications in the communities where sybil pain is most acute, so that Court of Humanity is operational and proven when the broader demand arrives.

1.12.2 Adoption Tiers

Adoption follows the pain gradient: communities already suffering from sybil attacks and willing to tolerate mechanism friction adopt first; mainstream users follow as the crisis deepens and proven alternatives exist.

1.12.2.1 Tier 1: Crypto-Native Applications

Crypto-native users understand bonds, tolerate complex UX, and already suffer from sybil attacks at scale. This community validates the mechanism and generates the first proof points.

Sybil-resistant token distributions. Projects lose hundreds of millions of dollars annually to sybil farming of airdrops and rewards. A distribution run through Court of Humanity that can publicly demonstrate resistance to farming — quantified savings, zero successful sybils — provides a concrete, dollar-denominated proof of value. Each airdrop that competitors lose to bots is an implicit case study for Court of Humanity.

DAO governance. The first DAO treasury drained through a sybil-manipulated vote will drive every other DAO to seek alternatives. The Democracy Launchpad toolkit targets small-to-medium DAOs where bond sizes are manageable, governance consequences are real, and the community is sophisticated enough to evaluate the mechanism.

Quadratic funding. Sybil manipulation of matching rounds (e.g., Gitcoin) has been a persistent and well-documented problem. A CoH-integrated funding round that demonstrably resists collusion rings is immediately compelling to the public goods funding community, which is vocal and influential in the broader crypto ecosystem.

1.12.2.2 Tier 2: Trust-Dependent Communities

As sybil attacks cause visible harm beyond crypto, communities whose core functions depend on trust will seek solutions. These users may not be crypto-native but are motivated enough by acute pain to adopt a new system.

Local civic governance. Participatory budgeting, neighborhood council votes, co-op decisions, and union elections are moving online and are trivially gameable. These processes are geographically concentrated (aligning with court coverage), the stakes are personal, and the current tools offer no sybil resistance. A successful pilot — for example, a participatory budgeting cycle in a Wave 1 court city that produces a legitimate outcome while a comparable city's process is manipulated — is a powerful demonstration. Court of Humanity's geographic court model is a natural fit: local governance users are already in the same jurisdiction.

Verified-human content and media provenance. As deepfakes escalate from novelty to crisis — fabricated political statements, impersonation scams, synthetic evidence — the ability to prove that a real, unique human created or endorsed a piece of content becomes essential. Court of Humanity can provide a simple standard: content signed by a bonded identity carries a machine-verifiable "verified human" marker. This integrates with Bonafide natively and can be adopted by any platform via the attestation interface. The value proposition sharpens with every deepfake incident.

Trusted reviews and ratings. Review systems on major marketplaces are widely understood to be compromised. A review platform where every reviewer holds a bonded identity and every review is disputable creates a premium trust signal that both consumers and businesses value. This is a consumer-facing application that requires no crypto literacy to understand: "these reviews are from real people who have something to lose if they're lying."

1.12.2.3 Tier 3: Mainstream Refuge

When sybil attacks affect ordinary users' daily experience — when social media is majority-bot, when scam calls are indistinguishable from real ones, when online votes are routinely manipulated — the general public will seek alternatives.

Human-only social spaces. Bonafide itself can serve this role. In a digital environment saturated with synthetic content and bot accounts, a social platform where every account is bonded and verifiably unique is not a feature — it is the product. The intentionally minimal feature set becomes an advantage: the platform's value proposition is authenticity, not engagement mechanics.

Anti-scam communication. Deepfake voice and video scams will escalate as the technology improves. A communication layer where participants can verify each other's bonded, biometrically verified identity becomes essential for high-stakes conversations — financial, legal, familial. The bond serves as a trust signal: the person you are speaking with has economic skin in the game tied to their real identity.

Human-verified dating and community admission. Bot-infested dating platforms and online communities are already driving user frustration. A platform where every profile is bonded and catfishing carries a real economic penalty — the bond is slashed on a successful dispute — offers a premium experience that users are willing to pay for. Typical dating app subscription fees already exceed the bond sizes required for social-class applications.

1.12.3 Strategic Principles

Build for the crisis before the crisis. The worst time to build an identity system is when everyone urgently needs one. Infrastructure, courts, verifier pools, and reference applications must be operational before the demand spike. This means absorbing the bootstrapping cost on crypto-native use cases while the mainstream crisis develops.

Let failures be the proof. Every sybil attack that breaks a competitor platform, every botted governance vote, every fake review scandal, every deepfake crisis is a proof point for Court of Humanity. The project should publicly document these failures and maintain clear comparisons with how CoH-integrated alternatives performed under the same conditions.

Anchor early applications to court geography. The first non-crypto applications should target communities within Wave 1 court jurisdictions. A civic governance pilot in Philadelphia, Austin, Los Angeles, or San Francisco maps directly to existing court infrastructure, verifier pools, and dispute capacity. Geographic alignment between the application's user base and the court's jurisdiction minimizes travel friction and maximizes verifier availability.

Optimize for the "relief" moment. The first time a user enters a CoH-integrated space — a social feed where every account is real, a review platform where ratings are trustworthy, a vote where every participant is unique — they should feel immediate contrast with their experience elsewhere. That emotional shift is worth more than any technical explanation of the mechanism. Onboarding should be designed around producing this moment as quickly as possible.

Grow islands of trust. Each CoH-integrated application is an island of trust in an increasingly untrustworthy digital environment. As conditions worsen, the islands become more valuable and attract more users. Eventually the islands connect: a user bonded on Bonafide can attest to a DAO, a review platform, and a civic governance tool without re-enrolling. The ecosystem's value grows super-linearly with the number of integrated applications.

1.13 Conclusion

Court of Humanity combines private biometric uniqueness with an explicit economic deterrent enforced by decentralized arbitration. Bonafide, the system's social identity layer, provides the observable context that makes the dispute mechanism practical: by requiring users to maintain persistent bonds and public profiles, it gives disputers the signals needed to confidently identify sybils and raises the detection probability D across the entire ecosystem. Downstream applications inherit this high D through zero-knowledge attestations of Bonafide account standing, enabling modest bond sizes without sacrificing security. By giving developers a measurable cost of forgery, we make sybil attacks economically irrational in practice. Fees on bonds and disputes accrue to stakers, sustaining a high security budget. The result is a reusable identity primitive suitable for digital democracy, authentic social identities, and machine-verifiable media, without relying on unverifiable hardware trust or fragile heuristics.

1.14 References

"Kleros Yellowpaper." n.d. https://kleros.io/yellowpaper.pdf.