HUMAN Protocol

Technical Whitepaper

Tokenizing Human Labor

Version 1.1 - DRAFT
January 28, 2021

Overview

The HUMAN Protocol is a broadly applicable approach to organizing, evaluating, and compensating human labor. It is designed to enable a new generation of machine intelligence to apply human labor to self-improvement in order to achieve human parity in task performance.

Today this work is commissioned by machine learning practitioners. The protocol's immediate application is thus to improve the most labor intensive problems in machine learning: making datasets fit for training via annotation and validating model inference quality.

While the HUMAN Protocol supports and improves today's practices, it is engineered for the next evolution of human inputs to machine intelligence: letting machines ask people directly for the data they need to improve.

HUMAN Tokens ("HMTs") serve as the medium of exchange in the HUMAN Protocol. They are EIP20-compatible tokens, and the complete system forms a decentralized platform with an open protocol. Each component receives a fee for its role, and interactions are coordinated via smart bounties on the Ethereum blockchain.

Status

MVP reference implementation is complete, and the HUMAN Protocol's first app (hCaptcha.com) is live on the Ropsten testnet, serving many billions of requests per month. Development has been underway since 2017, and open source implementations of many protocol components are already becoming available for different use cases. For example, Intel's open source CVAT tool was ported to run on the HUMAN stack in December 2020.

Background: the hCaptcha.com app provides a drop-in replacement for reCAPTCHA, and demonstrates the scalability of the underlying HUMAN technology. It equally provides a platform to displace Mechanical Turk, Figure Eight, and other intermediaries for some types of tasks. Requesters compensate website owners ("miners") based on challenges answered and computed accuracy. HMT mining power is thus directly proportional to human labor.

A Guide to Reading This Document

The core ideas of the HUMAN Protocol are broadly applicable and quite abstract, so we have chosen to write this overview with frequent examples from a particular application (hCaptcha.com) to demonstrate each component's role within that context.

Table of Contents

Team

The team behind HUMAN has decades of software and ML expertise, with a focus on meta-learning and visual domain ML at scale. Contributors come from Stanford, MIT, Apple, Google, Cloudera, etc. and have substantial ML, cryptography & security expertise, including previous work at Brave during the BAT ICO.

We have built large scale distributed systems for the web, for machine learning, and in finance. Many of the components used in the initial HUMAN Protocol reference implementation were written and validated at scale in our production codebases prior to being open sourced.

Value Proposition

HUMAN Protocol

Machine learning is currently applied to perhaps 1% of the problems it could efficiently solve.

This is in part due to the underlying friction in current workflows. Machines need human inputs, but human beings should not be the ones manually selecting and curating what is required to reach a particular accuracy target. Machines should be able to ask people and other machines for the data they need in order to improve and maintain inference quality.

This requires a programmatic interface. The HUMAN Protocol is an attempt to define that interface. It creates mechanical specifications of work and requirements that may be transmitted alongside escrowed funds that are released only upon successful completion of tasks. All managerial roles in the system can be performed by software: requesting specific work, evaluating the quality of that work, compensating workers, and delivering results.

There is an immediate need to improve the state of play by allowing more actors to participate and reducing friction on both sides of the review market. There is also a long-term need to enable the next generation of automated feedback systems for continuous improvement via human review. This is what the HUMAN Protocol is intended to enable.

hCaptcha.com

A substantial portion of dataset value is today captured by Google at very low cost via reCAPTCHA. Creating economic incentives for website owners by providing a drop-in replacement for reCAPTCHA will democratize access to high volume human evaluation.

This system tests for bots as well as reCAPTCHA while at the same time paying website owners for their audience, and serves as an implementation testbed in the HUMAN Protocol design-build cycle.

Architecture Overview

Requesters of work launch new bounties onto the blockchain that specify a job: the question to ask and the set of tasks to ask it about.

Exchanges pick up jobs, manage bidding on job types, and serve tasks to agents doing the work.

Recording Oracles collect potential answers and provide a rolling evaluation of answer quality.

Reputation Oracles make a final evaluation of answer quality and reputation score per job, and finally pay out bounties.

(See Definitions below for more details on each protocol component.)

Architectural Motivations

Why blockchain?

Many advantages today: allows “open books” to prove the system is fairly distributing bounties, enables efficient micro-payments, reduces required trust between protocol actors.

Even more advantages when blockchains are faster: verifiable reputation for every actor that opts in, oracle can compute earnings on-chain to further reduce required trust for interactions.

Why a new token?

Ethereum has the most robust smart contract support of any popular blockchain, but without additional development it is too slow and expensive for many applications.

Our Human Token contract thus implements a custom Bulk API that supports efficient micropayments via one-to-many bulk transfers. In other words, a single call can specify payouts from a smart bounty to 1000 addresses. This enables new and interesting use cases while remaining EIP20-compatible. We have open sourced the audited contract with a library and comprehensive test suite to help other projects in the wider Ethereum developer community adopt this approach as it suits their needs.

Today vs. Tomorrow

Current systems: no blockchain, centralized authority, REST API. No compensation for use.

Challenge: no blockchain today has adequate performance for use as a full-scale distributed human review system. Plasma, Lightning, Hashgraph, etc are still orders of magnitude away from necessary cost/speed performance and not yet robust. Future improvements may eventually make this feasible, but still early days.

Hybrid model thus ideal: faster to build, easier to scale using robust, proven strategies. Blockchain is used primarily for settlement, rather than trying to put every bit of logic into an on-chain oracle and every bit of data directly on-chain.

Design goals

Start more centralized, cheap and deterministic.
Build in ability to decentralize each component as tech evolves.

What do we want on the blockchain?

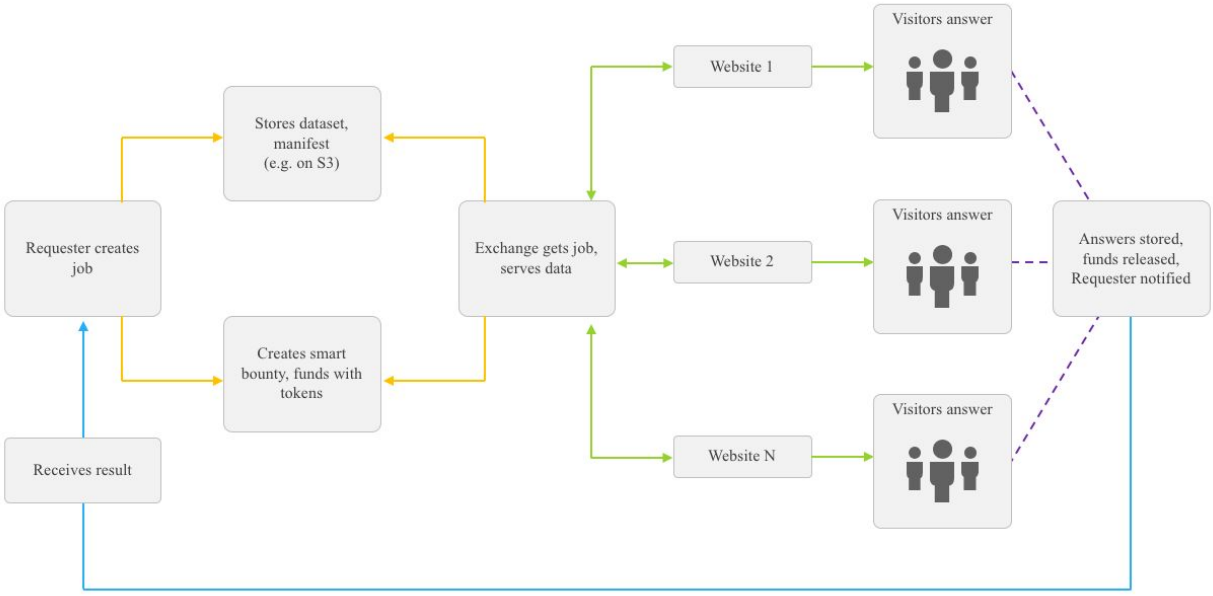
- Immediately:
    - Smart bounties: requester puts HMTs in escrow, describes their order, sets price.
    - Immutable hash, location (URI) of intermediate & final data, reputation results per job.
- Eventually:
    - Mining server trust scores
    - Requester trust scores
        - Neither one required for MVP. Can start with rolling reputation computation for graylists at the exchange level and mandatory request signing by reputation agent.

What do we NOT want on the blockchain?

- Data: images, texts, etc comprising datasets.
- Individual responses from mining clients: can be millions for a single job.
    - Side chains (Plasma etc) not ready for primetime, unfortunately.
- REST API services.
    - Mining client needs a location to hit, get go/no go response.

Example App Overview Diagram (Simplified)

An overview of the data path for hCaptcha.com, an application built on the HUMAN Protocol.



Definitions

Contract Originator

Software that generates the Smart Bounty (see definition below) for a given job. Responsible for launching new bounties into the blockchain for the requester, sending manifest off to Reputation Agent for signing, encrypting signed manifest. MVP: hosted by us behind REST API. Soon: open source.

Mining Server

Website that is serving hCAPTCHAs to its users, embedding a script hosted on an Exchange. https://exchange/mine?difficulty=3&safe_content=1

Mining Client

User of website who answers hCAPTCHAs embedded on website of mining server.

Exchange

Components:

- REST API gateway

- Blockchain monitor
- Job allocator

Serves up a script with hCAPTCHA functionality. (Generates JS with correct embedded vars from job.)
Keeps recent mining server stats, mining client request counts for graylisting, whitelist/blacklist; caches reputation scores computed by recording oracle and uses to stop bad actors early.
Watches blockchain for new jobs.
Chooses job from open bounties based on mining server criteria.
Fulfills mining client request with job data: URL to image, question, recording oracle server URL
Maintains sets of clients fulfilling Restricted Audience criteria, serves clients tests if necessary
If no jobs available: acts as captcha server using fallback.
Receives compensation based on requests served via smart bounty.
Publishes current bid prices for each task type, denominated in HMTs.

Recording Oracle

REST API gateway.
Receives mining client response, records response, returns yes/no (did the mining client succeed in their tasks?), weight, reputation update.
Has access to job data.
Each question gets asked N times. Pass weight starts at 0, then performs rolling computation of inter-rater agreement, confidence based on client + server reputation for agreement once N>2.
Stores final mining server result contributions in immutable form.
Updates blockchain with URI, hash once all results recorded to specified N per question.
Receives compensation based on requests served via smart bounty.

Reputation Oracle

Monitors blockchain for recording oracle updates.
Validates results against holdback validation set.
Updates blockchain to trigger release of escrow HMTs by smart bounty.
Updates exchange and mining server reputation scores based on accuracy.
Calls smart bounty function to distribute escrowed tokens to mining servers.

- For mining server earnings over threshold, immediate payment made.
- For micro-earnings too small to warrant spending the gas:
    - recorded in aggregate per-job earnings file, URI and hash stored in smart bounty.
    - swept in single transaction into holding account.
    - periodic transfers once micro-earnings hit threshold.

Receives compensation based on requests served via smart bounty.

Reputation Agent

Optional: signs smart bounty requests to apply its reputation to the request, validating that e.g. a human being has reviewed it, or that URLs are reachable etc. Validates manifests, returns signature to include in manifest. Called by Contract Originator.

Payment Agent

Optional: needed only to enable seamless onboarding for Exchange users with no native ETH addr. Creates ETH public/private keypair, returns public key address to Exchange to use as 'secret' in the reCAPTCHA lingo.

Requester

The requester of a human evaluation job, i.e. the machine learning model builder.

Smart Bounty

A smart bounty is a service contract initiated by the requester via the Contract Originator component that holds HMT in escrow. Contains:

- type of request: image labeling, etc
- number of answers sought per dataset point
- location of dataset manifest (URLs of images, question(s), etc)
- location of validation set manifest (URLs with answers)
    - readable only by reputation oracle
- bid amount
- expiration date
- valid exchanges for request
- valid reputation agent for request (exchanges may have their own whitelist)
- valid recording oracle for request (exchanges may have their own whitelist)
- valid reputation oracle for request (exchanges may have their own whitelist)
- whether content is "safe" for all ages (ignored until requester rep is >N)
- optional: valid mining servers, restricted audience
- optional: minimum mining server trust score

Below is a high level sketch of functionality. Please refer to the *hmt-contracts* repo's smart bounty example code and library for current implementation details.

Function: *jobIsValid(bool)* callable only by specified reputation agent.

- confirms that the job should be processed by an exchange: exchanges maintain their own whitelists of approved reputation agents, which can be derived from on-chain reputation scores.
- if false, bounty is canceled and requester receives balance in escrow, less fee.

Function: *storeIntermediateResults(URI or IPFS-style key, hash)* callable only by specified recording oracle.
- recording oracle encrypts results with reputation oracle's public key.

Function: *storeFinalResults(URI or IPFS-style key, hash)* callable only by specified reputation oracle.
- reputation oracle encrypts results with requester's public key.

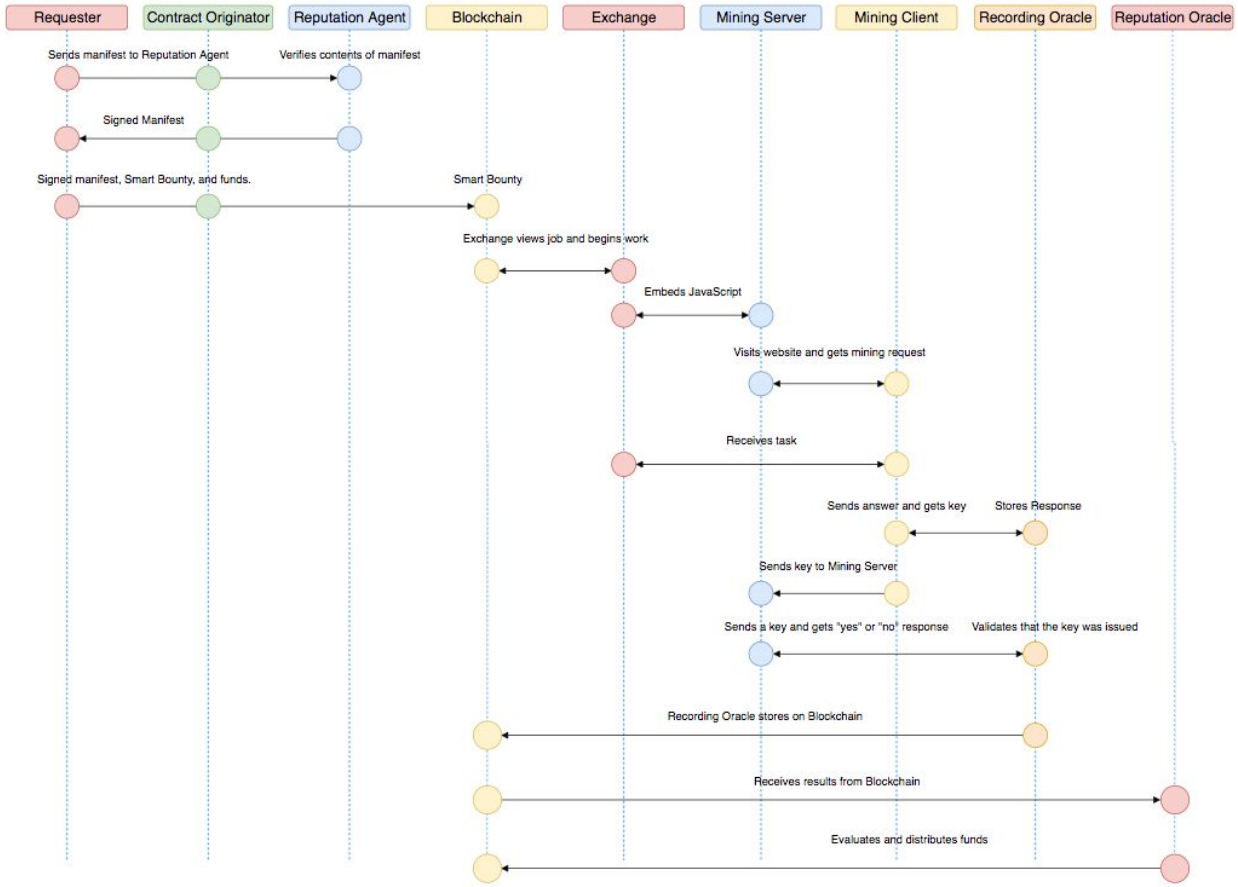
Function: *jobSatisfied(bool)* callable only by specified reputation oracle.
- triggers release of results to requester, payment of oracles and exchange

Function: *distributeBounty(miningserverAddress, amount)* callable only by specified reputation oracle.
- distributes specified amount of tokens from escrow to mining server each time it's called.

May be signed by reputation agent to prove some sort of validation has been performed on the request.

System Flow

Visualization of Detailed HUMAN System flow (batch mode)



Detailed HUMAN System flow (batch mode)

1. Requester creates smart bounty (service contract) via Contract Originator with HMTs in escrow.
    - Manifest optionally signed by Reputation Agent, vouching for requester without reputation; Contract Originator sends manifest to Reputation Agent via REST API.
        - Reputation Agent performs verification of manifest contents, e.g. runs images through adult content detection.
2. Exchange watches blockchain, picks up job.
3. Mining server embeds script from exchange.
4. Mining client:
    - Hits captcha request on mining server (website)
    - Gets dataset entry from location specified by exchange, i.e. image URL
    - Answers question, standard reCAPTCHA protocol follows:
        - Client hits POST on request, passes 'g-recaptcha-response' to mining server

- Mining server sends request to recording oracle
    - 'secret' (ethereum key in our case)
    - 'response' (from mining client)
- Mining server gets go/no go from recording oracle ('success' field in response)

5. Recording oracle stores results of mining server contributions per job in immutable form.
6. Reputation oracle evaluates results, makes determination about whether to release funds.
7. Reputation oracle calls smart bounty function *jobSatisfied(true)* to release results to requester, pay oracles and exchange.
8. Reputation oracle calls smart bounty function *distributeBounty(addr, amount)* for each mining server to send tokens from escrow to mining server addresses.
9. Reputation oracle handles earned amounts too small to economically transfer:
    - MVP: Reputation oracle sweeps remaining aggregate of earned amounts too small to pay for individual transfers into offline holding account, writes out ownership data for periodic review and distribution once total earned by a mining server across jobs is above threshold.
    - Later: maybe Plasma when it's ready.

Detailed HUMAN System flow (online/streaming mode)

1. Requester calls Exchange REST API *reserveAPIKey*, gets API key to include in manifest.
2. Requester creates smart bounty (service contract) via Contract Originator with HMTs in escrow, effectively putting a deposit down on future API requests.
    - Manifest includes reserved API key.
    - Manifest includes webhook endpoint controlled by Requester to receive streaming results.
    - Manifest optionally signed by Reputation Agent, vouching for requester without reputation; Contract Originator sends manifest to Reputation Agent via REST API.
        - Reputation Agent performs verification of manifest contents, e.g. runs images through adult content detection.
3. Exchange watches blockchain, picks up job.
    - API key is now active.
4. Requester uses API key to submit tasks directly to Exchange via REST API.
    - Exchange may perform additional verification of task contents, e.g. running images through adult content detection.
    - Exchange schedules tasks directly at current bid price for the task type to ensure lowest latency.
    - Exchange tracks total spend, expires API key when funds are depleted.
3. Mining server embeds script from exchange.
4. Mining client:
    - Hits captcha request on mining server (website)
    - Gets dataset entry from location specified by exchange, i.e. image URL

- Answers question, standard reCAPTCHA protocol follows:
    - Client hits POST on request, passes 'g-recaptcha-response' to mining server
    - Mining server sends request to recording oracle
        - 'secret' (ethereum key in our case)
        - 'response' (from mining client)
    - Mining server gets go/no go from recording oracle ('success' field in response)

5. Recording oracle stores results of mining server contributions per job in immutable form.
6. Recording oracle immediately forwards each response to Requester along with confidence score.
7. Reputation oracle evaluates final results, makes determination about allocation of funds.
8. Reputation oracle calls smart bounty function *jobSatisfied(true)* to release aggregate results to requester, pay oracles and exchange.
9. Reputation oracle calls smart bounty function *distributeBounty(addr, amount)* for each mining server to send tokens from escrow to mining server addresses.
10. Reputation oracle handles earned amounts too small to economically transfer:
    - MVP: Reputation oracle sweeps remaining aggregate of earned amounts too small to pay for individual transfers into offline holding account, writes out ownership data for periodic review and distribution once total earned by a mining server across jobs is above threshold.
    - Later: maybe Plasma when it's ready.

Basic Use: Today (Legacy Systems)

1. Create dataset of possible images, texts, etc.
2. Manually label some number of those images, reserve as validation set.
3. Upload to MT etc for labeling, object masking, etc.
4. Get label results.
5. Compare results to validation set to produce accuracy score, run again if below threshold.

Basic Use: HUMAN System (batch)

1. Create dataset of possible images, texts, etc.
2. Manually label some number of those images, reserve as validation set.
3. Create job:
    - choose a minimum accuracy score (enforced via validation set and inter-rater agreement)
    - upload dataset and validation set
    - place smart bounty at bid informed by current jobtype bid on chosen Exchange
4. Get label results from recording oracle or webhook delivery.

Basic Use: HUMAN System (online/streaming)

1. Reserve API key from Exchange.
2. Create job:
    - place smart bounty to spend up to N HMTs on Exchange.
    - specify webhook to receive streaming results.
3. Send API request to Exchange with individual task(s), datapoint(s).
4. Get label results from webhook delivery immediately.

Options

Proof of Stake vs. Proof of Balance

It is easy to fall into a cargo cult approach to proof of stake, adding it anywhere and everywhere simply to reduce token velocity. This tends to do a disservice to adoption by increasing friction, and often does not fit the real world dynamics of a protocol.

In our case, both the fundamental design and many different components work separately and together to prevent abuse and ensure it is not rewarded. However, as adoption grows it becomes more attractive to supplement confidence scores with other mechanisms.

Rather than risking staked tokens, we propose instead to treat the current HMT balance of the ETH addr requesting tasks to complete as a priority input when choosing which jobs and tasks to serve from the Exchange open order book. Holding more tokens will increase the probability of higher earnings due to better access to the highest bid jobs available on the exchange.

Similarly, we propose to sort equal bids from Requesters within the order book on the basis of HMT held by the requester address: holding more HMT gives higher execution priority to jobs launched by that account.

In this protocol's implementation, primary operation of the fairness algorithm that balances the increase of value received by existing vs. new network participants is devolved to individual Exchanges. We plan to release multi-agent simulation results to help understand this new model and its effects in various scenarios.

Factored Cognition

One area of research now finding applications in machine learning and other fields is *factored cognition*: decomposing more complicated work into its simplest cognitive components. Practical applications of this idea map very nicely onto the HUMAN Protocol.

Aggregating Labor Across Exchanges

For example, an Exchange may publish a custom high-level job type like "extract structured data from each page of a scanned form and return it in the schema I provide." This job has many sub-tasks: identifying text fields in the image, matching their type to the provided schema, reading the contents of the field, and so on.

Each of these sub-tasks can be mapped onto a standard job type provided in the initial Exchange reference implementation. That work can then be distributed across the entire network of Exchanges publishing those job types (based on real-time cost and reputation), and the sub-results combined to provide a final result.

The exchange fee mechanism has been designed to incentivize creation of these higher-level value-added job types, allowing Exchanges to set, publish, and vary their own fees based on market conditions.

Hierarchies of Trust

Many larger tasks can be decomposed into sub-tasks requiring very different levels of trust. Consider our document scan example above. Seeing the entire page requires high trust if it contains personally identifiable information (PII). However, once a field has been labeled by type then reading its contents may require no trust at all. A social security number has no value in isolation: it becomes PII only when combined with a name and address.

We can use this understanding to construct Exchange hierarchies that run the (often small) task portions that require trust into an appropriate venue, like a private Exchange with registered members. The results of that labor can then be used to generate trustless tasks to load into high volume and lower cost public Exchanges.

Restricted Audiences

A common desire in many forms of assigned labor is to limit the audience within a labor pool in some way. For example, a particular job may be intended only for people who are strong English speakers, people who live in the United Kingdom, or people able to pass a test containing similar questions.

Restricted Audiences are the HUMAN Protocol's solution to this requirement.

Simple language example

1. Requester defines a *restricted_audience* property in the job manifest. For example,
`"restricted audience": { "lang": [ { "en-us": { "score": 0.9 } } ] }`
2. Exchange restricts served tasks from this job to clients whose browser reports en-us at 0.9 or above.

User-defined task example

1. Requester specifies test in *restricted_audience* property of job manifest and required pass rate.
In manifest
`"restricted audience": { "tests": [ { "uuid": "uuid-goes-here", "uri":"https://test.com/1.json", "score": 0.9 } ] }`

In test JSON
List of tasks and questions, following standard manifest taskdata format. (See reference implementation in *hmt-contracts* repo.)
2. Exchange restricts served tasks from this job to clients who have passed the test at the specified score. Manifest is encrypted, keeping test data private.

Instant Onboarding

As demonstrated on hCaptcha.com: websites with pre-existing ETH accounts don't need to enroll anywhere to get started. They can simply drop in the JavaScript embed line and set their

ETH key on the server side. However, reputation oracles may choose to require registration on a whitelist prior to transfers in order to meet country-specific compliance restrictions.

Websites without ETH accounts can also get started almost immediately: after registering with an exchange they receive a temporary key that is in fact an Ethereum account managed by the Payment Agent (tied to their email address) and may enter their own ETH account at a later date.

Simplified Online Streaming Use Case

Allow "online" (streaming rather than batch) delivery to requester-controlled results endpoint with immediate payment. No reputation oracle analysis and no validation set required. Requires MVP attack mitigations described in Attacks.

Low Latency Online Requests and Results for Guided Model Discovery and Refinement

In some scenarios it is valuable to receive low latency human input on model inference quality to e.g. guide online learning or metalearning processes. When this behavior is required the requester can create a smart bounty with the job JSON specifying the "online" job type and a receiving server, prepaying tokens to the smart bounty. The nominated exchange provides direct REST API access for datapoint entry. All job datapoints submitted via REST API run at the highest bid price in order to guarantee queue priority.

Encryption of Intra-run Results

Results have low individual value and high aggregate value for this service. While all inter-component communication and final results are encrypted, it adds only minimal value to use e.g. homomorphic encryption on that data for intra-job storage. Not required for MVP, based on customer discussions.

Job Revocation

Exchanges and oracles need to monitor the blockchain for revocation notices. We currently have an `abort` function in the smart bounty, callable only by the requester, that terminates a job so long as no payouts have yet been made. In order to prevent abuse, a duration and job size-dependent cancellation fee is charged to the requester, paid to the exchange and oracles.

Signing Key Rotation: Signing Key vs ETH Key, Signing Key Lookup

Several entities in the system need to both sign/encrypt arbitrary data (of which a portion may be controlled by an attacker) and make contract calls to transfer funds.

Best practice is to rotate signing keys and avoid key reuse between arbitrary data signing/encryption and ETH transfers. We have designed and built a smart contract solution that may be interesting for the wider Ethereum community: an authenticated blockchain key-value store that allows arbitrary metadata to be uniquely and verifiably tied to an ETH addr.
(Please see the hmt-ethkvstore repo for more details.)

This allows us to provide a lookup layer between the ETH address of the agent or oracle (doesn't change) and the signing key or other metadata (may change).

This means that we start with agent security guarantees that are only as good as the underlying ETH addr key management, but additional safeguards and layers of defense are possible. For example, the public signing key can be published via multiple avenues and automatically checked, and any unexpected disagreement can alert the recipient parties to engage manual intervention.

Letting the Reputation Agent (Rather than Requester) Construct the Validation Set

We may not want to allow the requester to choose the validation set. Why? They might choose non-representative questions, when a random sample would work better. The reputation agent can provide a list of data to validate to the requester, or alternately send it for labeling at high confidence and simply let the requester decide whether those answers are good enough to proceed with job submission.

Reputation Scores

Overview

Reputation scores enable early detection of abuse, robust anonymous proofs of humanity, and other functions. MVP: the Recording Oracle keeps track of per-client and per-server reputation scores on the fly in order to assist in initial confidence computations. These are passed back to the Exchange along with pass determination, and final computations are made by the Reputation Oracle.

Initial Application: Early Abuse Detection

The Exchange can maintain a rolling cache of reputation scores received from the Recording Oracle in order to to aid in real-time abuse identification. At the end of a job the Reputation Oracle then computes authoritative client and server reputation scores on a per-job basis, using validation set data (if available) and the Recording Oracle's intermediate results.

Other Applications: Anonymous Proofs of Humanity

Once a robust reputation score and confidence value can be computed on a per-client basis there are many applications. One example is proofs of humanity. View & click fraud is rampant in the online ad market. Altering advertisers' payouts based on sampling of client and server reputation scores effectively creates a mechanism to punish sites serving a higher percentage of paid impressions to non-human bots, increasing ROI on ad spend as more humans and fewer bots receive paid impressions.

Reward Mechanism

MVP: Individual users of public exchanges (e.g. hCaptcha.com) already receive an effective benefit in that their user experience improves alongside increased reputation. Users of private exchanges can equally be rewarded by supplementing proof of balance computations in open order book sorting, job execution sequencing, and other mechanisms.

Attacks

Agent: Mining client

Attack: Bust captcha by sending nonexistent datapoint answers if communicating directly with recording oracle.
Mitigation: Recording oracle checks existence of job ID on blockchain, datapoint signature validates for job ID.

MVP Mitigation: Recording oracle checks existence of datapoint in cached job ID dataset manifest.

Attack: Nuke mining server reputation by providing wrong answers.
Mitigation: Client reputation score used to limit effect of bots.

MVP Mitigation: Mix-in standard CAPTCHA scoring to screen for bots.

Agent: Mining server

Attack: Fabricate garbage captcha requests/answers.
Mitigation: Reputation oracle runs periodic validation on results, sends mining server reputation to zero if accuracy threshold is not met. Mining server payout is a function of volume and score.

MVP Mitigation: Blacklist or whitelist of mining servers checked by exchange. Recording oracle can submit provisional entries to blacklist during job for obvious abusers.

Agent: Exchange

Attack: Steal HMTs by mining only known validation set answers.
Mitigation: Only reputation oracle can read validation set answers.

MVP Mitigation: Exchanges have high trust, are whitelisted.

Attack: Steal mining server HMTs by forging recording oracle, proxy mining client answers.
Mitigation: Recording oracle address is signed by requester, mining client checks signature. Recording oracle address may also be hashed into individual URLs of dataset.

MVP Mitigation: Exchanges have high trust, are whitelisted.

Attack: Nuke mining server reputation by fabricating garbage requests/answers.
Mitigation: This one is a bit tricky without letting the mining server sign answers. Recording oracle sees refer headers, but those can easily be forged.

MVP Mitigation: Exchanges have high trust, are whitelisted.

Agent: Requester

Attack: upload validation set guaranteed to be wrong to avoid paying full fee.
Mitigation: charge a fee for all requests high enough to discourage this, do not release results from recording oracle if accuracy target not met.

MVP Mitigation: pay-as-you-go streaming is not vulnerable to this attack.

Attack: upload dataset with some percentage of pornographic images marked as safe.
Mitigation: reputation agent runs images through IM's existing adult content detectors before signing job.

Attack: attempt to use control of plaintext to break keys of reputation agent (partial but not full control over signing of arbitrary plaintext), recording oracle (partial control over encryption of arbitrary plaintext), or reputation oracle (partial control over encryption of arbitrary plaintext).
Mitigation: generate, publish secondary keys for encryption to avoid any key reuse, use strong signing algorithms.

Agent: Reputation Oracle

Attack: hack reputation oracle, steal keys, declare jobs in progress complete and transfer tokens from smart bounty.
Mitigation: reputation oracle is strongly isolated, has minimal attack surface. Additional mitigations private.

Attack: hack reputation oracle, steal keys, transfer sweeps funds to attacker.
Mitigation: reputation oracle is strongly isolated, has minimal attack surface. Distribution occurs frequently to minimize size of sweeps, job earnings verification data is stored on blockchain for real-time analysis and anomaly detection. Additional mitigations private.

Additional Uses

Content Moderation

OCR Assistance

.. and many more. The Protocol allows arbitrary job types to be published by each Exchange.

Appendix A: Disclaimers

THIS IS NOT A PROSPECTUS OF ANY SORT
This document does not constitute a prospectus of any sort; it is not a solicitation for investment and does not in any way pertain to an offering of securities in either Canada or the United States, and Canadian and United States residents are expressly excluded from contributing in exchange for any Human Tokens in the public contribution offering. This document constitutes a description of the Human Token platform and the functionality of the Human Tokens; it is for informational purposes only and may change as the technology develops over time.

DISCLAIMER: This draft Human Token Technical White Paper is for information purposes only. Intuition Machines Inc, and all affiliated and related companies do not guarantee the accuracy of the conclusions reached in this paper, and the whitepaper is provided “as is” with no representations and warranties, express or implied, whatsoever, including, but not limited to: (i) warranties of merchantability, fitness for a particular purpose, title or noninfringement; (ii) that the contents of this whitepaper are free from error or suitable for any purpose; and (iii) that such contents will not infringe third-party rights. All warranties are expressly disclaimed. Intuition Machines Inc., and its affiliates expressly disclaim all liability for and damages of any kind arising out of the use, reference to, or reliance on any information contained in this white paper, even if advised of the possibility of such damages. In no event will Intuition Machines Inc., or its affiliates be liable to any person or entity for any direct, indirect, special or consequential damages for the use of, reference to, or reliance on this whitepaper or any of the content contained herein. Recipients are specifically notified as follows:

· No offer of securities: Human Tokens (as described in this Human Token Technical White Paper) are not intended to constitute securities in any jurisdiction. This White Paper does not constitute a prospectus nor offer document of any sort and is not intended to constitute an offer or solicitation of securities or any other investment or other product in any jurisdiction.

· No advice: This Human Token Technical White Paper does not constitute advice to contribute in exchange for any Human Tokens, nor should it be relied upon in connection with, any contract or purchasing decision.

· No representations: No representations or warranties have been made to the recipient or its advisers as to the accuracy or completeness of the information, statements, opinions or matters (express or implied) arising out of, contained in or derived from this White Paper or any omission from this document or of any other written or oral information or opinions provided now or in the future to any interested party or their advisers. No representation or warranty is given as to the achievement or reasonableness of any plans, future projections or prospects and nothing in this document is or should be relied upon as a promise or representation as to the future. To the fullest extent, all liability for any loss or damage of whatsoever kind (whether foreseeable or not) which may arise from any person acting on any information and opinions contained in this Human Token White Paper or any information which is made available in connection with any further enquiries, notwithstanding any negligence, default or lack of care, is disclaimed.

Risk warning: Potential contributors should assess their own appetite for such risks independently and consult their advisors before making a decision to contribute in exchange for any Human Tokens (HMTs).