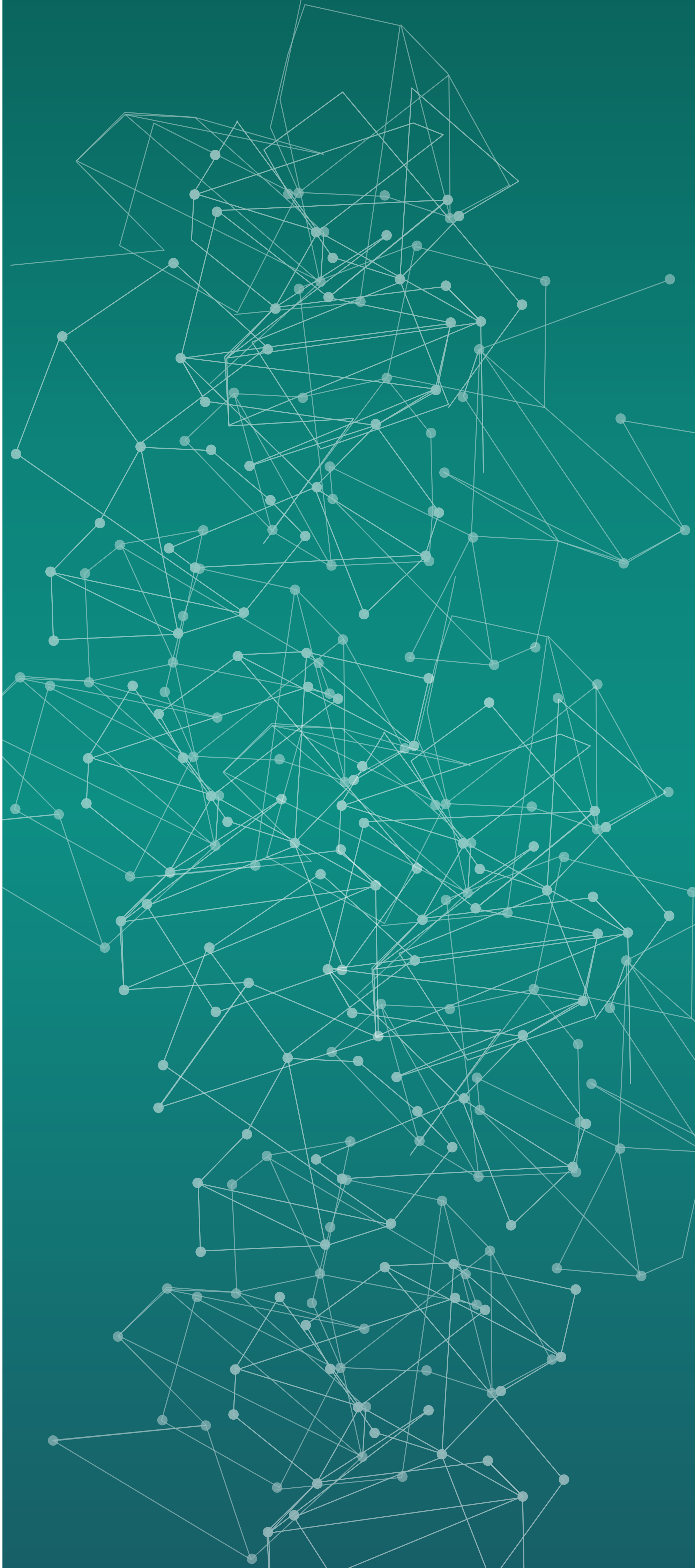
METACERT PROTOCOL

TECHNICAL PAPER



The MetaCert Protocol

The MetaCert Protocol (“Protocol”) will be an open security Protocol for the Internet storing trust and reputation information about Uniform Resource Identifiers (URIs) which includes domain names, applications, bots, crypto wallet addresses, Application Programming Interfaces (APIs), and content classification. The Protocol’s registry is machine-readable and queryable for use by Internet Service Providers (ISPs), routers, crypto exchanges, Wi-Fi hotspots, mobile devices, browsers, websites, and applications to help address cyber threats such as phishing, malware, brand protection, child safety, and news credibility.

This document outlines the technical specifications around the components that will make up the Protocol. From the technology and the usage of the META Token (the “Token”) and system architecture to the Protocol services and the processes that govern the system, each facet of the Protocol plays an important role in ensuring the stability and harmony between the network and the various participants that interact with it.

Technologies

The Protocol's decentralized URI classification registry will be built primarily using Node.js and Python with the following beneficial characteristics:

- **High-speed response.** The system will be able to process queries up to 2,500 transactions per second via a sync layer. This enables the platform to ensure real time transaction processing speeds.

- **Elasticity.** Blockchain technology ensures continued operation even if up to one-third of the nodes in the blockchain network are disabled.

- **Improved security.** Python was chosen because it is designed without segmentation faults which guarantees thread safety. By doing this, we ensure all thread processes occur on the platform in a robust manner.

The System

The Protocol is divided into **three independent parts:**

1. An end user participant system (codename: "ChainKit") using Python, Node.js, and Elasticsearch providing a user interface for parties to submit, validate and dispute URI classifications. These parties are participants in the Protocol and they include the following user classes: Submitters, Validators, Purchasers and End Users. The participant system provides authentication and profile management including (but not limited to) surfacing publicly available immutable metrics generated by the Protocol, such as reputation rating, Token metrics and submission or validation statistics.

2. A decentralized blockchain registry that is an independent, distributed storage system for all ledger transactions relating to URIs held within it that are governed by participant Token transactions using smart contracts.

Ethereum smart contracts will be used to enable the management of required entry conditions, digital ledgers, and Token mechanics. The smart contract is an independent intermediary between two parties (for example between the Submitter and Validator of the same URI classification submission). All actions processed through a smart contract require an agreement with a digital fingerprint from both parties who wish to participate in each transaction.

The Protocol implements the following key functions through smart contracts:

- Control of an “event ledger” recording all actions (based on HASH) that will be accessible to all Nodes on the network.

- Control of a URI’s “status” recording submissions, validations, and disputes of URI classification submissions.

- Control of “attributes” relating to Token credit or debit for participants.

- Control of “metrics” relating to participant reputation.

A smart contract is constructed after successful validation of a submission. After a parameterized waiting period without a dispute being raised against the validation, a Token payment allocation is distributed to users involved in Protocol participation.

In the event of a dispute of a URI classification, an auxiliary smart contract is initiated between the disputer and the Validator(s) that requires token participation to execute. This ensures the system remains open for critical review while discouraging objections from bad actors.

3. A synchronization system (codename: “HyperChain”) using Python, Node.js, and Elasticsearch that provides a layer between the blockchain registry and the ChainKit. This system is responsible for interacting with the blockchain to eliminate latency issues with write transactions. To ensure high performance, we have chosen to architect microservices (described later) utilizing a RESTful API, auto scaling and multi availability zone clusters.

Types of Data

There are three primary types of data across the system network:

1. **Proprietary Data:** This is data controlled by a Node Operator and is generally unique to that operator.

   Example: If Cisco wanted to ensure their Submitters and Validators on their Node were registered Cisco users.

2. **Regulated Data:** This is data where accessibility is limited generally due to privacy constraints. This data may not be unique and Nodes may choose if it is shared between them.

   Example: The participant age verification for validators classifying pornography URIs, DNS records or DNSSEC Records.

3. **Common Data:** This is data that is open for general use on the blockchain. This type of data has no restrictions on its usage.

   Example: Transparent event information showing who submitted, validated, or disputed any URI classification submission along with all the URI's ledger history showing reputations of all participants involved in each event.

High level Ecosystem Overview

The decentralized architecture of the Protocol is designed to allow for specific rules to be applied to URI classification types.

For example, imagine the need for different parameterized rules to be applied to phishing classification compared to pornography classification. Any Node Operator can add a classification type of URI at a Node level when using the Regulated or Common data type as described in Data Types section above.

Each process has two common components. We'll call them (1) **base components** and (2) **extended components.**

Base components are standard rules, processes and data points which cannot be altered or overwritten. Think of these as laws common to all Nodes.

Extended components are additional data points on top of base components that Node Operators may choose to add and are always specific to those Nodes. Think of these as optional bylaws for each Node.

Each event requires an encrypted code and signature to verify each Protocol participant, which then forms a unique transaction ID (also known as a universally unique identifier or UUID).

Each unique transaction enters the blockchain via the HyperChain, and the Protocol is able to present a record of the current submission queue in order to process it in real-time. This is because blockchain technology usually takes time to verify each transaction written to it. The blockchain is reevaluated every 2,016 blocks, which is why the process takes time. For the Protocol we need a speedier presentation of activity hence the production of the HyperChain layer to provide this service, which is especially important for real-time sensitive classifications such as phishing.

Each transaction on the Protocol's decentralized blockchain registry will always be public and verified to avoid problems of authenticity and duplication. Protocol Nodes are responsible for verifying the authenticity of each transaction that includes information such as a new or disputed URI submission, their resulting validations and the reputation of participants.

The Protocol is designed to prevent bad actors from gaming the system by having a waiting period called **the challenge period** which must expire without a dispute being made against the validation of a submission. This process is built-in before Token amounts are paid out to participants for URI classification transactions.

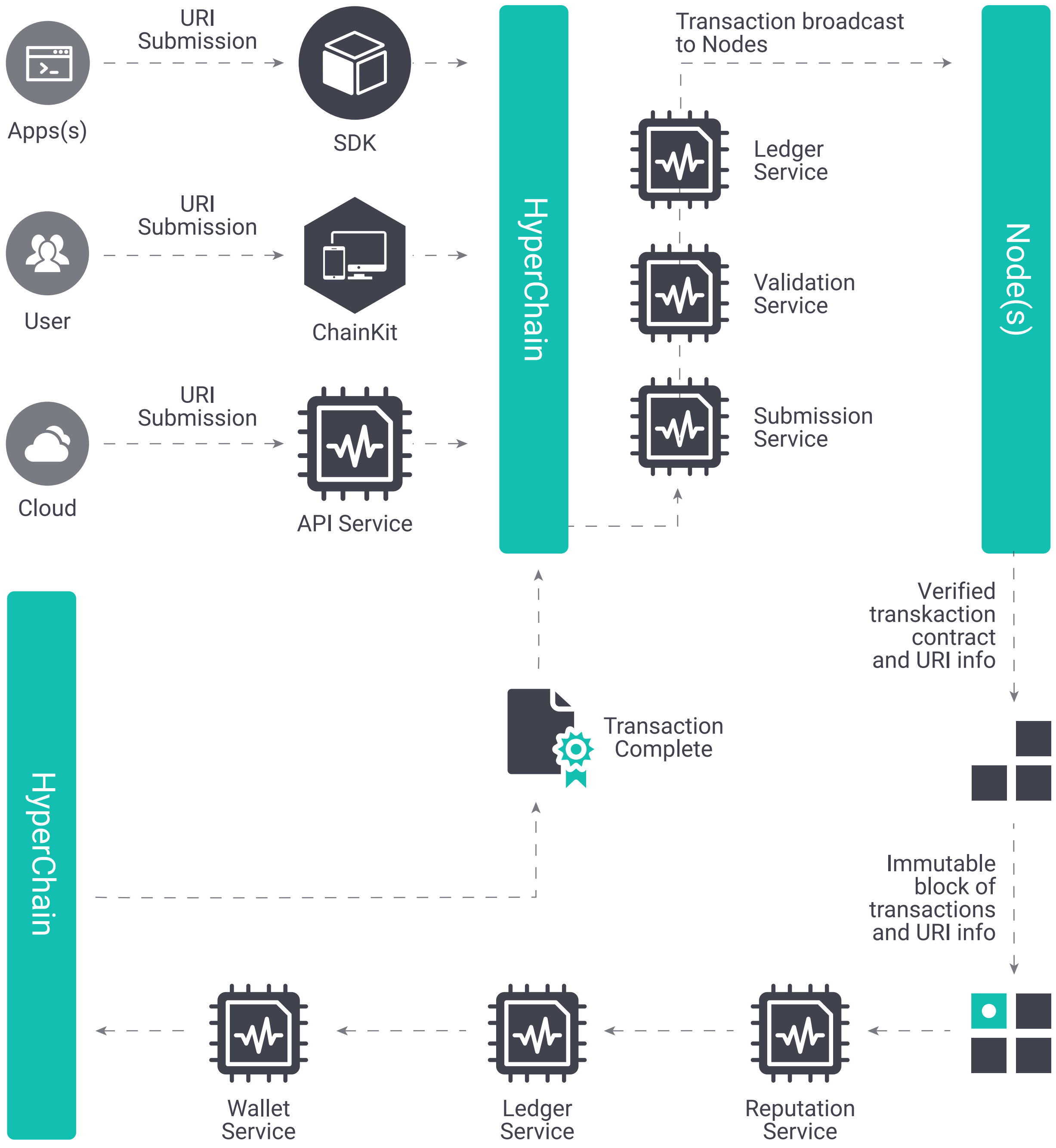
To prevent bad actors from disrupting the Protocol, participants have a maximum of 5 active new submissions in the challenge period at any given time unless their Protocol reputation status permits this limit to be removed.

System Architecture

In the system architecture, the blockchain occupies the secure base layer to document consensus on data bindings of the state of digital assets (e.g., domain verification or URI classification). We chose to follow a design principle to keep as much complexity and logic outside of the blockchain layer as possible.

Think of the system as clusters of high speed virtualization sync Nodes wrapped around existing blockchain capabilities to enable real time transactions at the participant level, just like virtual machines are constructed on top of physical hardware in traditional Software as a Service (SaaS) models.

The primary benefit of our virtual sync Node implementation is better fault tolerance in case of a failure of the underlying blockchain.



Protocol Services

Architecturally, all services are built independently of each other, which increases the stability of the entire Protocol. The disruption of one service will not cause the failure of the another or the system at large.
All components of the platform interact with each other via a RESTful API, specially designed to provide data exchange between the internal and external services of the system.

Synchronization Service

The Protocol uses the HyperChain, a service that interacts with the blockchain and the ChainKit platform, functioning as a high-speed sync of the blockchain's ledger. Thanks to this service architecture, even its complete disconnection will not affect data continuity and safety.

Consumers of Sync Service:

- MetaCert (Super Consumer): Unrestricted, uninterrupted, unthrottled access.

- Enterprise Consumer: Predefined or Prearranged, throttled access.

- Generic Consumer: Limited access granted to developers and entities seeking to build applications around the Protocol. Restriction level may be flexible based on usage.

API Service

The API service is an essential component of the ChainKit, which will operate independently and allow access not only at the transactional level but as well as the data level. This in turn empowers developers and any entities interested in the implementation of distributed ledger technology. The API will have features such as an unconfirmed transaction confidence factor, dependable WebHook or WebSocket-based events, on-chain microtransactions, and a query for ledger event metadata.

The API can be integrated into popular applications (Slack, Firefox, Chrome) and be used to build your own tools and services.

Submission Service

Submission Flow

A Submitter must first be authenticated with a participant account before being able to submit a URI. The submission flow will vary depending on the Submitter class defined below. The standard Submitter class submission flow is:

- Submitter enters or pastes URI for classification.
- Selects one or more classifications to attribute to the URI.
- Chooses if the classifications should apply to either:
    - the entire domain.
    - the current URI directory (folder) and below.
    - the specific URL only.
- Some classifications, such as ownership verification may require
- Token charges while other URI classifications require Token to be staked in order to submit.
- Token charge or staking fee for submitting a URI is shown on the submission page.
- Submitter pays and receives notification of Token charge if required.
- Confirmation view displays estimated time for verification based on current backlog.
- When verification is completed, a notification is sent to the Submitter via their chosen channel of communication defined in their profile settings.

Submitter Classes

Based on the Submitter class, a different submission and validation flow will be adopted. The Protocol will initially cater to the following Submitter classes:

- **GTLD Registrar:** Domain registrar that wishes to bulk submit current or future domains for classification. It is not necessary that these domains contain any content.
- **Expert:** Industry experts, who are well known for their knowledge and expertise in one or more categorization types.
- **Domain Owner:** Domain/Site owners who wish to claim their domain classification.
- **Standard:** General community Submitters.
- **Pro:** A general Submitter that has achieved high enough reputation and has concurrent submission limitation increased.

Submission Status

Based on Submitter Class and category submissions in the Protocol will be associated with one of the following statuses:

- **Accepted:** When a URI submission is accepted, but is not in review.
- **Rejected:** When a URI submission is rejected and tagged with a rejection reason.
- **In Review:** When a URI submission is Accepted and available for review by a Validator.
- **Pending:** When a URI submission is accepted but the DNS does not yet resolve.
- **Claimed:** When a domain URI for ownership is validated (note: this status has an expiry date).
- **Expired:** When a domain URI that has been validated for ownership reaches its expiry date.
- **Validated:** When a URI submission is reviewed and validated by 1 or more Validator.
- **Disputed:** When the combination of a URI and classification is in the process of being challenged by a participant, this status remains until a resolution is reached.

Submission Acceptance

Submission acceptance will occur following predefined, automated acceptance checks:

- DNS resolves correctly (For Domains)
- Domain Control Validation (For Domain Owner Claim)
- Reverse Proxy Validation (For URI Submissions)
- TLD/GTLD Validation (For XXX Category Submissions)
- Metadata Crawling (For XXX Category Submissions)
- URL Resolve Validation (For All URIs)

Ledger Service

The Protocol's Ledger Service is more than a shared, immutable ledger for recording the history of transactions. It provides a permissioned network with known identities. Transparent and highly available, the Ledger Service is designed to record all events, from the request of URI submission to a participant's reputation, since events are cryptographically linked to one another via timestamps and other attributes.

The Ledger Service will allow URI submissions, validations, disputes, participant reputations, and Token transactions that are open, transparent, and verifiable.

Authenticity Service

The Protocol's authenticity service acts as gatekeeper, to other micro services. The primary function of this service is to provide a boolean response plus timestamp of a queried event for other microservices that need confirmation of a classification for a URI.

The service will provide responses for events such as Ledger Service encryption and security, transaction authenticity validation, user identification and more.

This service will employ industry standard encryption to secure transactions to and from this service module.

Reputation Service

The reputation service is designed to calculate ratings for all participants (Validators, Submitters, disputers and consumers). The rating algorithm is based on multiple key metrics that dictate user reputations and credibility scores, which include:

- Positive Metrics:
    - Account Age
    - Positive acknowledgement by other participants weighted by their reputation rating
    - Number of followers within the Protocol
    - Number of successful transactions (Submissions/Validations)
    - Number of votes cast on the winning side of a dispute resolution
- Negative Metrics:
    - Negative acknowledgement by other participants weighted by their reputation rating
    - Number of unsuccessful submissions
    - Number of non-concurred validation judgements
    - Number of votes on the losing side of dispute resolution
- Ordained Metrics:
    - Granted to existing industry experts that fulfill registration criteria issued by the Protocol

The entire reputation system is governed by a core base rule, which cannot be modified by any Node. However, an extended rule can be incorporated onto the base rule with a certain degree of limitation on a Node level. Each reputation gain and loss is written to the blockchain, thus making the entire reputation ledger decentralized, i.e. reputation data resides on chain, however additional data from extended rules will always be held off chain for use within the Node only.

ChainKit Service

To manage submissions, validations, and update Nodes, developers will have access to all the necessary tools for the ChainKit in the form of API documentation, and Software Development Kits (SDKs) for popular platforms and software suites.

SDK Services will help developers make their app integration process as smooth as possible, allowing them to flexibly adopt Protocol events into their workflows for submissions, validations and data consumption.

Additional Microservices

A large number of additional services will be implemented on the Protocol's Nodes. Of note, the development of submission management using Artificial Intelligence ("AI"), which will use Big Data analysis modules and an algorithm to collate additional information after submissions. This will provide an opportunity to automate the filtering of unresolvable or restricted access URI submissions in the system.

Another example would be for the AI to obtain analytical data on the queries on the ledger to monitor the frequency of use for each URI on the Protocol to determine popularity.

Processes Overview

URI Submission

A user encounters a phishing site and decides to warn others about it. If they are not already a participant in the Protocol, they sign up to the ChainKit and go through the new URI submission process and propose a classification as phishing for the URI. The ChainKit will then determine whether the URI is a deep linked individual page, domain path level (i.e. folder), entire domain or subdomain as intelligence for validation.

If the URI is already classified as phishing in the Protocol, the participant will be informed and the process will not proceed.

Note: *New URI submissions do not require a Token amount to be staked for classification. However, some classifications such as ownership do.*

The submission goes into a review queue within the ChainKit and its status is changed as available for validation. Once a single Validator confirms the proposed classification, parameterized Token amounts are transferred to a temporary wallet for the Submitter and Validator.

A smart contract will be constructed containing all the events, from submission to review queue and the eventual validation outcome only when a parameterized number of Validator(s) confirm or reject the submission.

When the challenge period has expired without a dispute being raised, the smart contract will be completed and the Token amounts transferred from a temporary wallet to the Submitter and Validator(s) involved.

URI Validation

A Validator receives notification that there are submitted URIs for them to review within categories they have chosen and earned enough of a reputation score to review. A Validator's personal review queue of URIs within the ChainKit is randomly assigned.

The URI intelligence that the ChainKit provides to help a Validator form their classification validation decision does not include any Submitter information.

When their decision is made, they participate in the same process described above in the URL submission process.

A Validator can accept or reject a URI submission in their queue or they can choose not to make a judgement on it. There is not an option to remove a validation request. If a Validator chooses not to make a judgement on a URI classification it goes to the next available randomized Validator.

The queue will have a maximum parameterized amount of URIs (i.e. 10) for review at any time in any one Validator's queue. When their queue is empty, a validator can request another batch of URIs for review up to the maximum.

If they need a break from validating URIs, a Validator may set their participant account in the ChainKit to pause once they have cleared their current queue.

If a Validator has not signed into the ChainKit for a parameterized period of time and have not viewed their queue, the system will reallocate their URIs for review to another Validator.

As an extra measure to prevent bad actors gaming the system, the Protocol ensures that the same Validator cannot approve the same Submitters entries above random statistical coincidence.

For almost all categories, a Validator cannot approve their own submissions unless a category's parameters allow it, i.e., domain ownership classification.

Reputation Change

Every participant in the system starts with zero reputation. A participant's base reputation value is computed based on their ChainKit account activity. Some of the variables that contribute to the reputation algorithm include account age, URIs submitted, validation outcomes, network usage of URIs, and disputed classifications in favor or against.

Once a sufficient base reputation has been achieved, a participant's reputation can level up to become a special class such as an "Expert", when consensus is reached within the Protocol system which determines at what score leveling up can occur.

Reputation score can be negatively impacted whenever a submission classification proposal is not validated or when a validation is overturned by a dispute claim. Reputation can also be affected for a dispute if the dispute claim is repealed. Each of these events and associated smart contracts forms a series of ledger entries.

It's possible for a participant to generate a negative value reputation score, losing their ability to participate in the Protocol, and losing any in-process Tokens held in a temporary wallet in the Protocol.

The exact mechanics of the algorithm for reputation will be published publicly and iterate over time to ensure continuous improvement and transparency on this critical element of the Protocol.

Dispute a Classification

If a dispute against a URI classification is made by a participant in the Protocol, an event in the ChainKit initiates a smart contract between the Disputer and the first Validator of the classification. This smart contract contains the details of the claim and dispute.

A challenge fund is created for the dispute and held in a temporary temporary wallet within the Protocol. The outcome of the dispute will determine how the challenge fund is distributed.

Process Mechanism

Submission and Validation Mechanism

To increase the quality of submissions and ensure bad actors cannot game the system to their advantage, the URI submission and validation service employs some base submission and validation rules which can be extended by category owners for the categories they own. Base rules are applied to general as well as to specific categories to guarantee transparency as well as a set of standards that governs the system.

URI Submission Pre-check Processes

When a submission is received, the system performs a series of pre-checks as time saving measures, including added benefits during the validation process. These pre-checks include:

- **Is the URI classified?** This is a basic check to ensure a submission hasn’t already been classified
- **Does the URI resolve?** If a submitted URI is a web URL and it resolves, it will be placed into the general validation review queue. Otherwise, it will be marked in the system as “Awaiting Confirmation” for an expert Validator to review

Following these initial pre-checks, the system crawler will collect meta-data from the URI.

Metadata Collection

The system crawler harvests information about a website and its contents. This process allows for additional pre-checks to help aid the classification of a submission. For example, if the metadata contains phishing or pornography keywords, the system would record this in the submission's intelligence record to be used when performing assisted validation.

Additional Pre-checks for Pornography (XXX) URIs

The system performs a few additional pre-checks for submissions to identify XXX URIs, which are:

- **Does the URI contain the .XXX TopLevel Domain (TLD)?** The system checks to see whether a submission uses the .XXX TLD
- **Is the domain name string classified as XXX?** The system checks to see whether the domain name is already in the registry and classified as XXX
- **Does the URI contain any XXX images?** The system utilizes the Amazon Web Services (AWS) Image Recognition API to identify potential XXX images.

If any of these specific pre-checks are true, the system will update the submission's intelligence record, which can be used in the assisted validation process.

URI Validation Methods

The Protocol has two types of validation methods for URIs: Assisted and Peer.

Assisted Validation Process

During the submission precheck process, if the URI validation intelligence record contains data, a Validator may opt to request for an assisted validation. This means that they do not need to wait through a designated holding period (7 to 14 days depending on reputation score of the Validator) following validation to receive their Token reward. This is because the validation is considered stronger with the system having intelligence on the URI. Should a Validator select this option, they will be required to share a percentage of their Token reward with the system.

If the URI is submitted by an expert Submitter with a high reputation score, the smart contract will be completed and the validation will automatically be marked as “Validated” in the system.

Between the submission and validation period, a dispute claim can be filed, which will withhold the validation until the dispute is resolved.

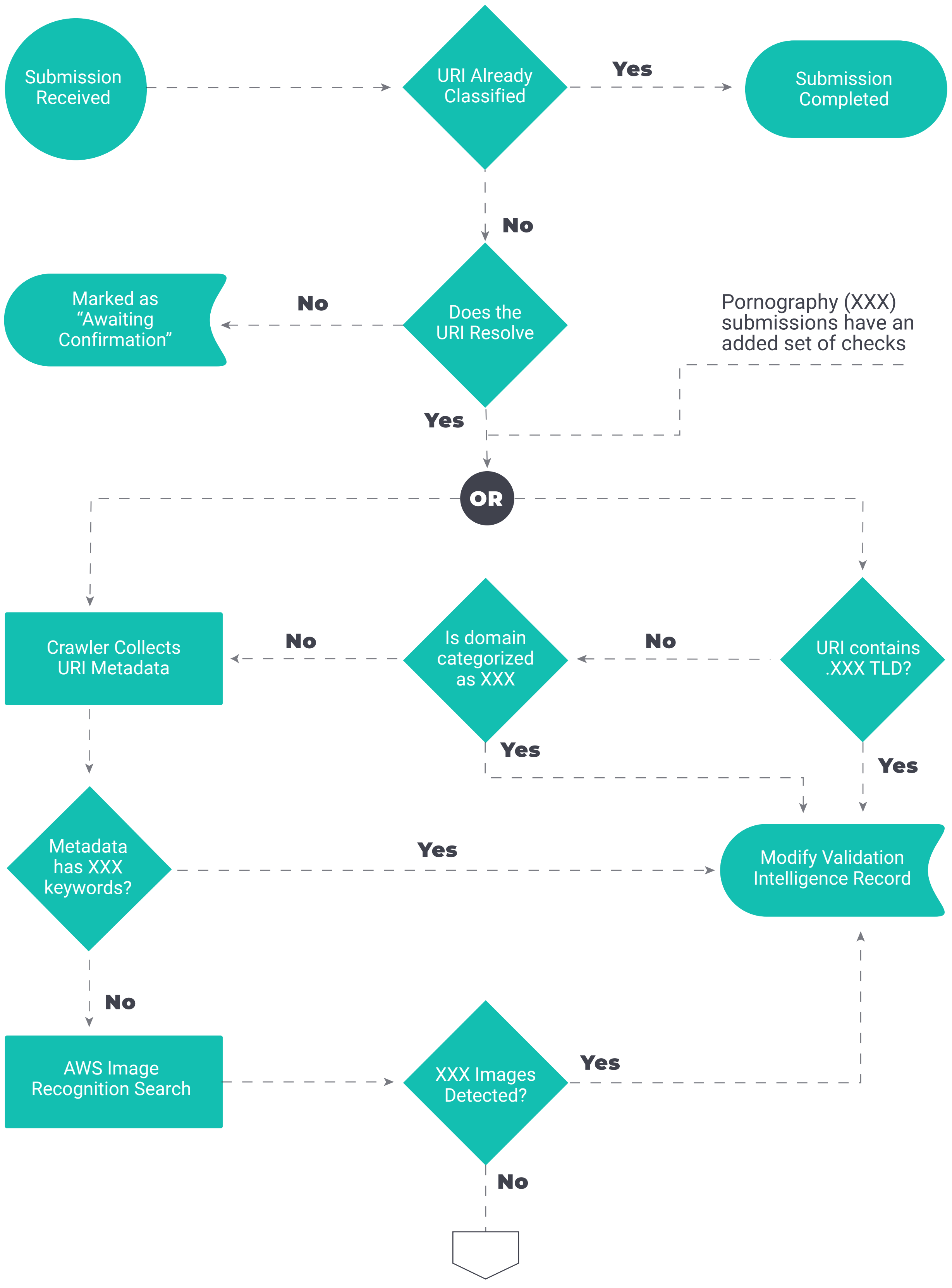
Peer Validation Process

If the system is unable to provide any intelligence data during the pre-check process, the URI submission is placed into a review queue. From here, multiple Validators (minimum of two) can reach an agreement (consensus) on the submission claim. Once consensus is reached, the status of the URI changes to “Validated” and a smart contract is constructed. There is a designated holding period (7 to 14 days depending on reputation score of validator) before Token reward is distributed to the Submitter and Validators. During this time, a dispute claim may be filed, adding an additional delay and a disputing party to the process.

Peer Endorsement for Validation (Optional)

A Validator may review a subset of validation records and agree or disagree with the results, which will affect the weighted adjustment to the validation process. Regardless of the outcome of the validation, the Validator will receive a percentage of Token for their work.

Overview of Assisted Validation for XXX Content


Submission Received
URI Already Classified
Yes
Submission Completed
No
Marked as "Awaiting Confirmation"
No
Does the URI Resolve
Pornography (XXX) submissions have an added set of checks
Yes
OR
Crawler Collects URI Metadata
No
Is domain categorized as XXX
No
URI contains .XXX TLD?
Yes
Yes
Metadata has XXX keywords?
Yes
Modify Validation Intelligence Record
No
AWS Image Recognition Search
XXX Images Detected?
Yes
No

Ownership (Brand Protection)

URI Submission (Pre-check)

The existing pre-check processes apply to brand ownership submissions including checking to see if a URI is already classified or whether or not it resolves with a few notable differences.

Notable Differences

- **Token Share for Submitter:** One of the key aspects of the brand ownership submission process is that the Submitter pays for the validation
- **Metadata Collection Differences:** Depending on the type of ownership submission received, the system will collect different sets of metadata, which includes:
    - Domain ownership will result in the collection of a domain's age and ownership information
    - Social account ownership will result in the collection of the account's status (public or private), account age, locale, activity and followers and friends
- **System notification to domain and account owners:** Efforts are made to contact the owners by email (for domain ownership) or social media message (for social account ownership) to inform them of a pending validation request

URI Validation by Owner

Similar to the general validation process, there is an assisted validation process and an optional peer endorsement process for both domains and social accounts.

Assisted Validation Process

Domains

- Domain owners must pay Token to initiate validation
- Owners can verify ownership by uploading a file to their domain or adding a CNAME record to their DNS records

- Information is recorded to the ledger including but not limited to:
    - Whether the owner responded to the email notification
    - Reputation score for the Submitter
    - Reputation score for the owner, especially if they are an existing Validator
    - Search engine rankings
    - Screenshot of the domain/website
    - All inbound domain links to the homepage including their classifications
    - Whether or not the domain has an entry on Wikipedia
    - Historical data from the Internet Archive's "Wayback Machine"
- A smart contract is constructed

Social Accounts

- Owners send a private or direct message requesting a validation link
- A private link is sent to the owner via private or direct message to complete the ownership validation process
- Information is recorded to the ledger including but not limited to:
    - Whether or not the owner responded to the private or direct message.
- A smart contract is constructed

Dispute Claim Process

In order for a dispute claim for a classification to proceed, the dispute must come from a registered participant in the Protocol, and they must place a Tokenized amount as a stake that is a parameterized multiple of the original validation earnings amount. Optionally, the original Submitter and Validators may place their own optional defense stake within parameterized time period in order to defend the dispute. After the parameterised time period elapses, a smart contract is constructed for the dispute.

The dispute is agreed only when (a) one or more Validators agree with the dispute or (b) one or more Validators disagree with the dispute. Certain categories require more Validators to agree or disagree with the dispute. For example, phishing submissions require a minimum of five validators to agree or disagree. Once a dispute claim is agreed upon, the smart contract will be completed and the URI will be declassified from the category it was assigned to.

Further work and discussion will be carried out with the governance body to determine what should happen if a dispute is not agreed upon or if a dispute is not defended by the original Submitter or Validators.

The Token reward for Disputer or Validators will be released 15 to 30 days after declassification assuming there is no disagreement (negative vote) during this timeframe. If there is a disagreement, the dispute is reopened and the cycle continues until majority consensus is established after another 15 days.

However, if the dispute is upheld, the original Submitter and Validator will lose reputation points, and any Tokens they may have staked as a challenge to the dispute will be forfeited.

Decentralized Technical Governance

The Protocol needs to be governed in a way that there are no central points of failure or control and various stakeholders can have representation in the Protocol development and direction.

The Protocol will undergo periodic hard forks to upgrade core functionality and to add or update Protocol operations.

Not all software updates are consensus breaking and therefore don’t require a hard fork. Upgrades that do need a hard fork, however, are harder to deploy and require (a) community agreement over Protocol changes, and (b) software updates across all deployed Nodes.

The Nodes that don't upgrade will be disconnected from the main network and end up on a forked network. If the community has a disagreement over Protocol changes, this usually leads to a fork into separate networks. Currently, the Protocol does not have an algorithmic metric for community consensus and Protocol changes but we will work towards establishing one through discussion with the whole community before they are activated.

The Protocol will introduce an algorithmic metric for community consensus based on the economic distribution of the Token. Token owners can sign messages from their respective owner addresses to participate in a voting mechanism. Different proposed Protocol changes can have different thresholds for activation.