Harnessing Collective Wisdom: A Trust Propagation Model for Decentralized Networks

Author: Harshil Dave @hersheylasers

Summary:

How can you quantify public confidence in Smart Contracts? I flip the identity question, rather than being concerned with the identity and reputation of individuals, I would like to ask, do you trust Gitcoin passport? Orbis? Newcoin? Disco? I present an analysis of a trust propagation algorithm designed to compute the 'confidence' or trustworthiness of claims and identities using attestations and the trust scores of attesters. The algorithm considers a network of attestations from various users and entities, aiming to offer a straightforward confidence metric for claims. Using Python, we simulate a network with honest, dishonest, and trusted actors to evaluate the algorithm's efficacy. Ultimately, our algorithm produces 'confidence scores' suitable that can be associated with the safety and reputation of smart contracts or on-chain organizations. A practical example is harnessing collective wisdom of attestations to identify safe smart contracts or web3 services and assign objective 'Safety Confidence Score' to smart contracts. *A crude description of this product is 'Google Reviews for Smart Contracts.'* However, our approach takes into account a whole network of attestations, and I also show that personalized scores can calculated based on your preferred trust sources.

Section 1: Introduction

Calculating and Propagating Trust in Decentralized Networks

The concept of calculating and propagating trust metrics in decentralized networks is not new. In the early 2000s, EigenTrust emerged as a solution to calculate the trustworthiness of seeders in decentralized peer-to-peer file-sharing networks. The primary goal was to reduce the risk of spreading malware from untrustworthy sources.

For our project, we've chosen to adopt the algorithm proposed by [Grüner et al]. This decision was influenced by the algorithm's heavy reliance on attestations and its compatibility with the Ethereum Attestation Service (EAS) and Ethereum itself. The algorithm is structured around three main components: an attester, a claim, and the receiver (attestee). The calculations, as detailed in Section 2, allow the attester to propagate its trust score to the claim, which then gets transferred to the receiver. In a decentralized setting, other participants can also make attestations to support the attester, a specific claim, or the receiver directly. This system is designed such that honest actors can achieve high trust scores, and true claims can garner high confidence scores. While this approach captures the collective wisdom, it doesn't offer the granularity and provability of systems based on zero-knowledge proofs. However, the overhead to run this system is low, since it does not require special knowledge, software, or complicated procedures by participants. It only requires that participants make attestations on EAS, which are simply signed messages in particular schemas. The algorithm can be run autonomously on a decentralized compute platform, such as Chainlink.

To maintain transparency and avoid potential conflicts of interest, our service will not engage in making any 'claims'. We won't be involved in creating attestations about someone's human-ness or the safety of a smart contract. Our primary role will be to develop the algorithm for trust propagation, deploy on Chainlink, and continuously make attestations about the confidence score attached to each claim, attester, and attestee. However, we would need to onboard an initial group of trusted attesters to bootstrap the network. For the use-case of smart contract reputation, we would like to gather attestations from auditors, such as those on Code4rena or Immunefi.

Openness and Customizability

The algorithm we're presenting is open source. While we might be the first in implementing it and making attestations on EAS, other providers can also access the public data, run the algorithm, and make their own attestations. In fact, the community can even make confidence/trust attestations about our service, reflecting their level of agreement or disagreement with our findings. The success of this approach would be determined by it’s adoption for determining the trust of smart contracts and on-chain entities.

Furthermore, the algorithm is designed to be customizable. Users can adjust parameters to give additional weight to friends or other trusted sources. They can then run the calculations to generate custom confidence scores for all interconnected claims, allowing for a dynamic and continuously updated trust landscape. With an improved frontend and UI/UX, this app can allow for users to create their own trust network. However, in my opinion, most users will adopt the default trust model for a use case such as smart contract trust.

Section 2: The Quantifiable Trust/Confidence Model

Foundations of the Model

At the heart of our trust propagation algorithm lies the principle that trust is not static but dynamic. It evolves based on the interactions and attestations within a network. The model we've adopted from [Gruner et al] is designed to quantify this trust, transforming subjective judgments into objective, numerical confidence scores.

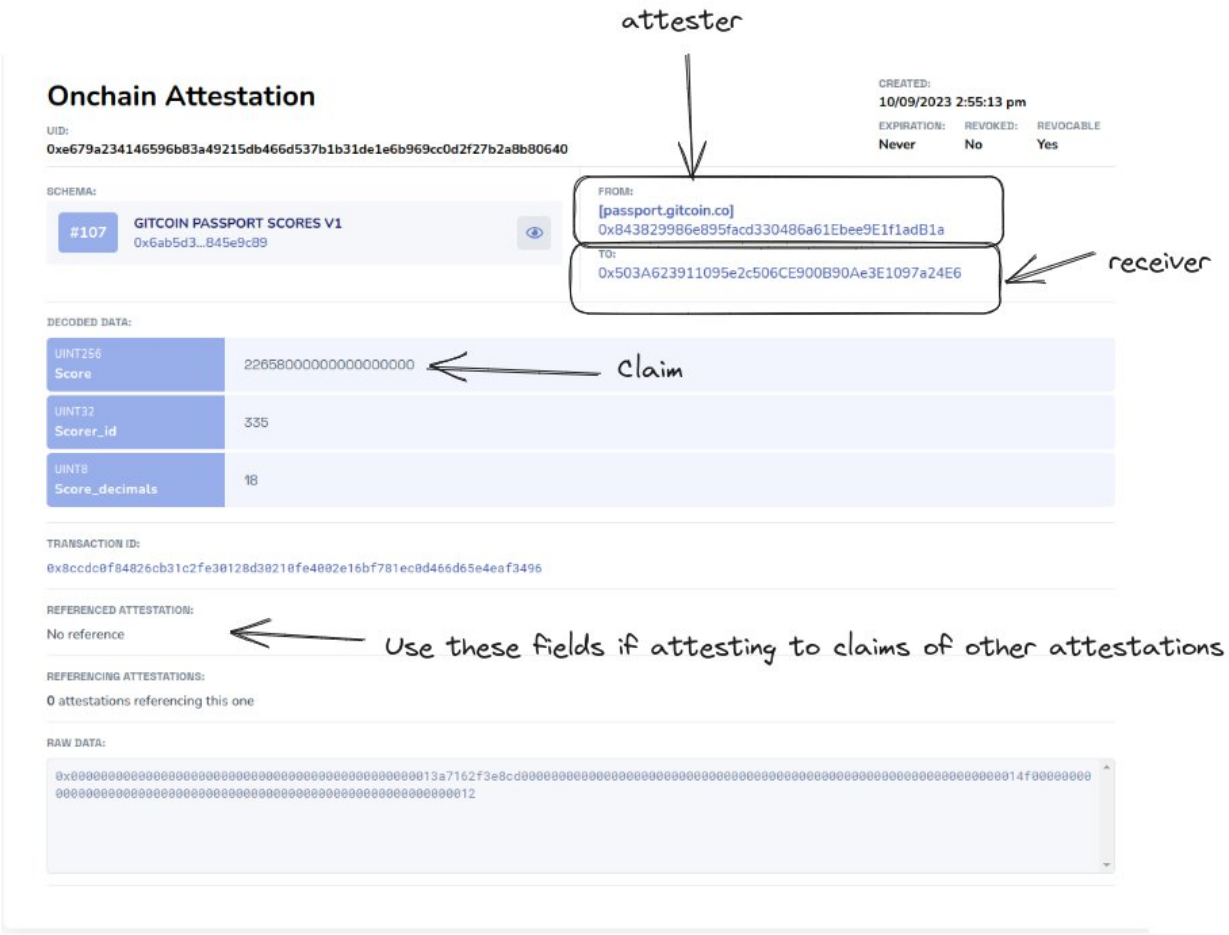

Figure 1: Example attestation and the components of this algorithm with a Gitcoin score attestation

Key Components

1. **Attester**: The entity that makes a claim about another entity or itself.
2. **Claim**: The statement or assertion made by the attester.
3. **Receiver (Attestee)**: The entity about which the claim is made.

Trust Propagation Mechanism

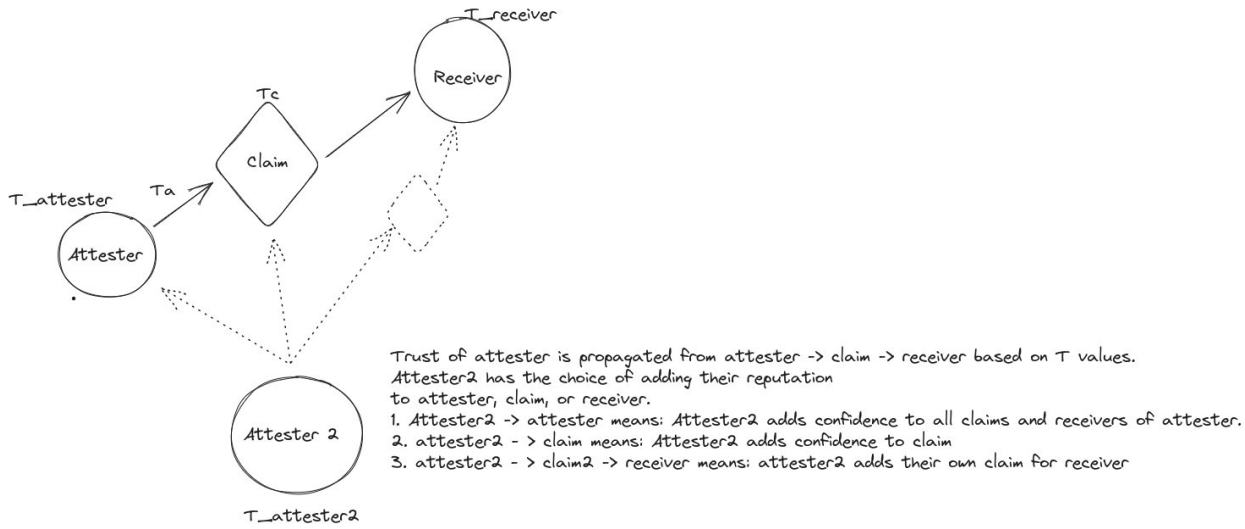

Figure 2: Diagram of the attesters and claims

The algorithm operates in a cascading manner:

1. **From Attester to Claim**: The trustworthiness of the attester influences the confidence in the claim they make. An attester with a high trust score will lend more credibility to their claims.

2. **From Claim to Receiver**: The confidence in a claim subsequently affects the trust score of the receiver. If a claim about an entity is deemed highly confident, that entity's trust score is positively impacted.

Incorporating the Network Effect

In a decentralized environment, the trust dynamics become more intricate. Multiple entities can attest to the same claim or receiver, leading to a web of interconnected trust relationships. Our algorithm takes into account:

- **Direct attestations:** Claims made directly about a receiver.

- I**ndirect attestations:** Claims made about other claims or attestations.

- **Collective wisdom:** The aggregate trust derived from multiple attestations, capturing the 'wisdom of the masses'.

Mathematical Representation

Please refer to the [1] for the equations and details. The main equations are copied here for convenience. In short, we calculate Ta (Trust of attestation) -> Tc (Trust of Claim) -> Ti (Trust of identity) in order. In Tc and Ti, we include all of the attestations linked to the claim or identity. This way, multiple attestations about a claim makes the claim stronger, and multiple claims about the receiver makes the confidence in the receiver stronger.

$$T_A : [0,1) \mapsto [0,1)$$

$$T_A : (t_i) \mapsto \begin{cases} t_i \cdot d, & t_i \geq \epsilon_{T_A} \wedge (c,i) \notin CI \\ 0, & t_i < \epsilon_{T_A} \vee (c,i) \in CI \end{cases}$$

$$T_C : (t_{a_1}, \ldots, t_{a_n}) \mapsto 1 - e^{s \cdot ((\sum_{i=1}^{n} t_{a_i}) \cdot n)}$$

$$T_I : \mathbb{R}^+ \mapsto [0,1)$$

$$T_I : (u_i) \mapsto \begin{cases} \frac{1-e^{k \cdot (u_i - f)}}{1+e^{k \cdot (u_i - f)}} \cdot 0.5 + 0.5, & u_i > 0 \\ 0, & u_i = 0 \end{cases}$$

The figures below show how the Tc and Ti metrics evolve as number of attestations grow. Notice that the values are always between 0 and 1. This allows us to easily understand the scale of trust, as well as allow trust to grow with a large number of positive attestations.

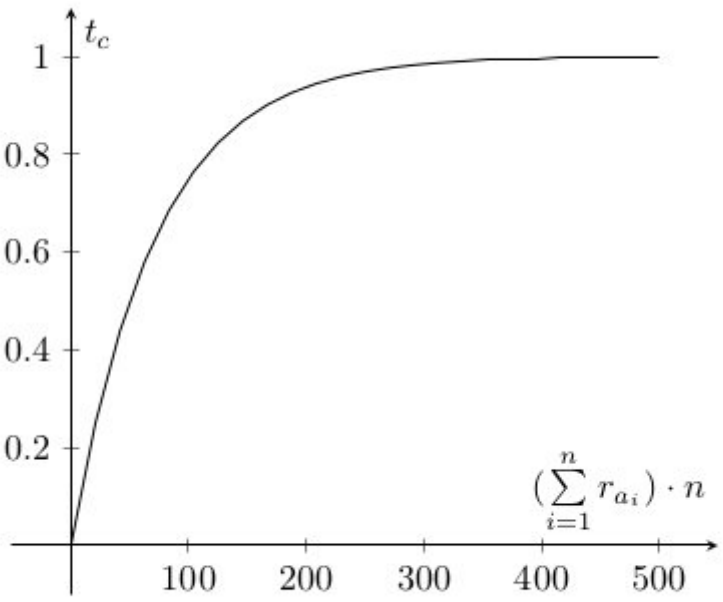

Fig. 3. Graph of T_C

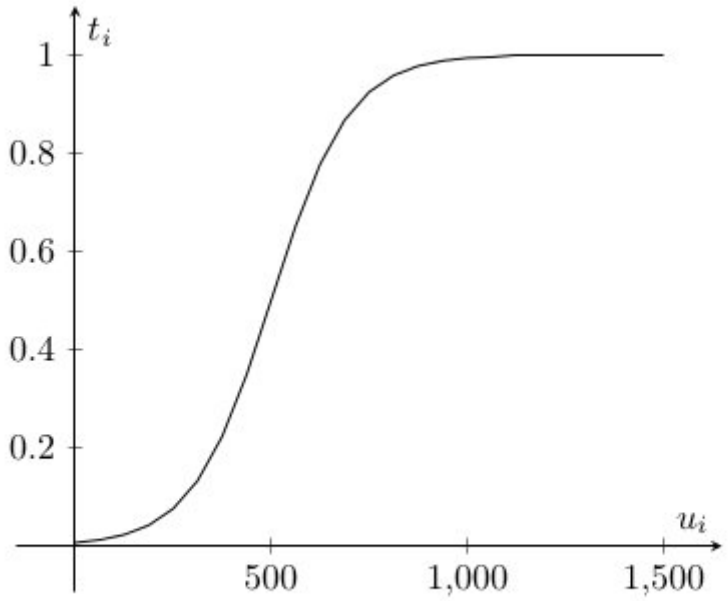

Fig. 4. Graph of T_I

Ref [1] uses the following description to subjectively describe trust/confidence levels:

• No Trust ($0 \leq ti \leq 0.2$).

• Limited Trust ($0.2 < ti \leq 0.4$).

• Medium Trust ($0.4 < ti \leq 0.6$).

• High Trust ($0.6 < ti \leq 0.8$).

• Superior Trust ($0.8 < ti < 1$).

Section 3. Analysis of the Model

Attestation Generation

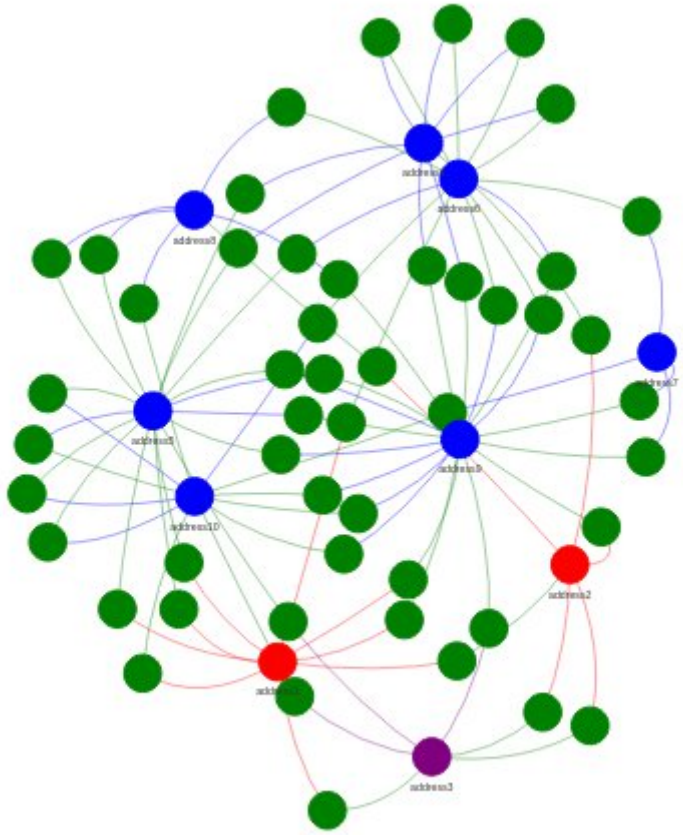

Figure 3: Example of generated attestations. Blue points are honest actors. Purple points are Trusted actors. Red points are dishonest actors. Green points are attestations/claims

The foundation of our trust propagation model is the attestations. These are the claims made by one entity about another, and they serve as the primary data points for our algorithm. To simulate a real-world scenario, we've developed a mechanism to generate fake attestations. This involves:

Role Assignment: Each entity (or wallet address) is assigned a role - honest, dishonest, or trusted. This role determines the nature and trustworthiness of the attestations they produce.

Claim Generation: Depending on their role, entities generate claims about others. For instance, an honest entity will produce accurate attestations, while a dishonest one might produce misleading claims randomly.

Trust Score Initialization: Every entity starts with a baseline trust score of 0. Trusted entities (shown in purple) are given a trust score of 1. As attestations are made and processed, these scores get updated.

The calculate_trust Function

Central to our model is the calculate_trust function. This function processes the generated attestations and updates trust scores accordingly. Here's how it works:

Trust Propagation: The function begins by propagating the trust from the attester to the claim and then to the receiver. This is based on the principle that the trustworthiness of an attester influences the confidence in their claim, which in turn affects the trust score of the receiver.

Network Effect: The function takes into account all attestations related to a particular claim or receiver. This includes both direct and indirect attestations. By aggregating these, it captures the collective wisdom of the network. We do this by calculating trust recursively. After some iterations, the trust scores of all entities converge to a solution.

Dynamic Updates: As new attestations are added or existing ones are removed, the calculate_trust function can be rerun to update the trust scores dynamically.

Analysis and Results

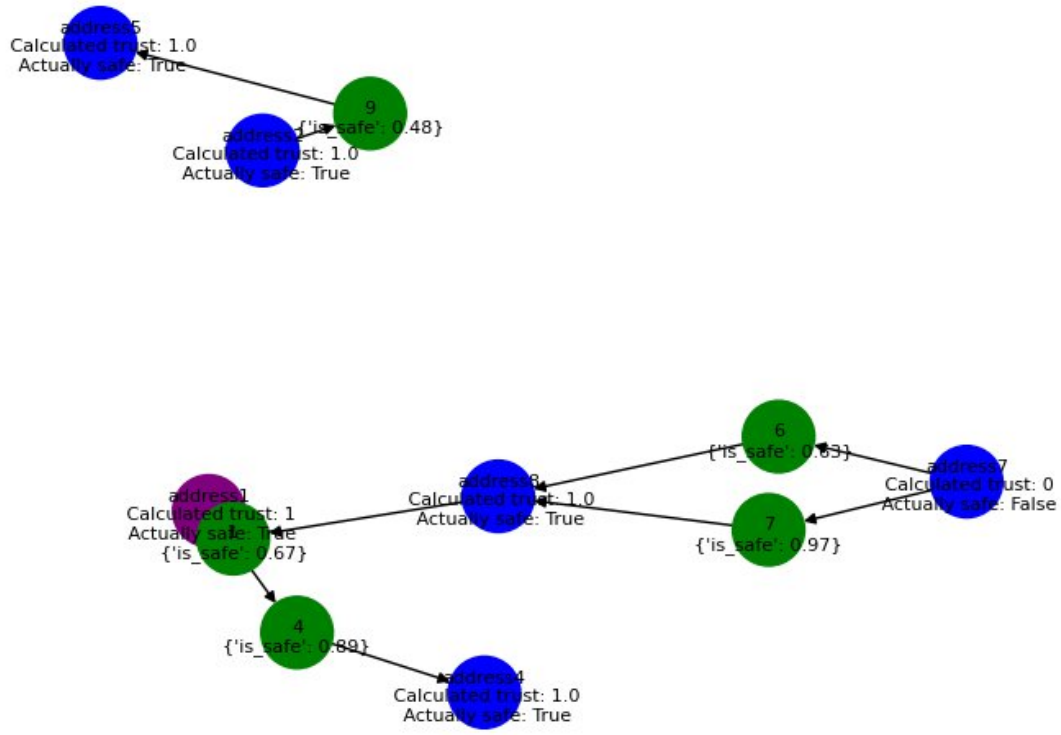

Figure 4: Example data. Blue nodes are honest actors, but some are given low 'safe' scores, indicated by the string 'Actually Safe: False'. Purple are trusted, which are automatically given Calculated Trust = 1. Green are attestations. All claims in this example say 'is_safe' about each other. Only a subset of all nodes are shown for clarity.

Figure 4 shows example results on the graph. Notice that the 'Calculated Trust' score matches the assigned roles correctly. For example, the 'Calculated Trust' of all 'Actually Safe: True' are 1.0, while the 'Calculated Trust' of the node on the right with 'Actually Safe: False' is 0. Note that untrusted Red nodes were filtered out in this example, but they also fail to garner any trust score.

The algorithm appears to work as intended. As an example, we generated 100 users (wallet addresses). We assign 20% to be dishonest, 20% to be trusted, and 60% to be honest. With 1000 attestations, it took about 2-3 iterations for the algorithm to converge, which took < 5 seconds on my home pc.

The end result was the majority honest actors had close to 100% trust and dishonest actors never got above ~2% trust. Note that based on the coefficients chosen in the equations, it takes ~500+ attestations to achieve a confidence score of >0.9 for users. This is shown in Figure 5, which shows the calculated trust of each wallet, as the number of total attestations grows. It is strange, however, why some wallets experience a sudden increase, or drop, in trust score.

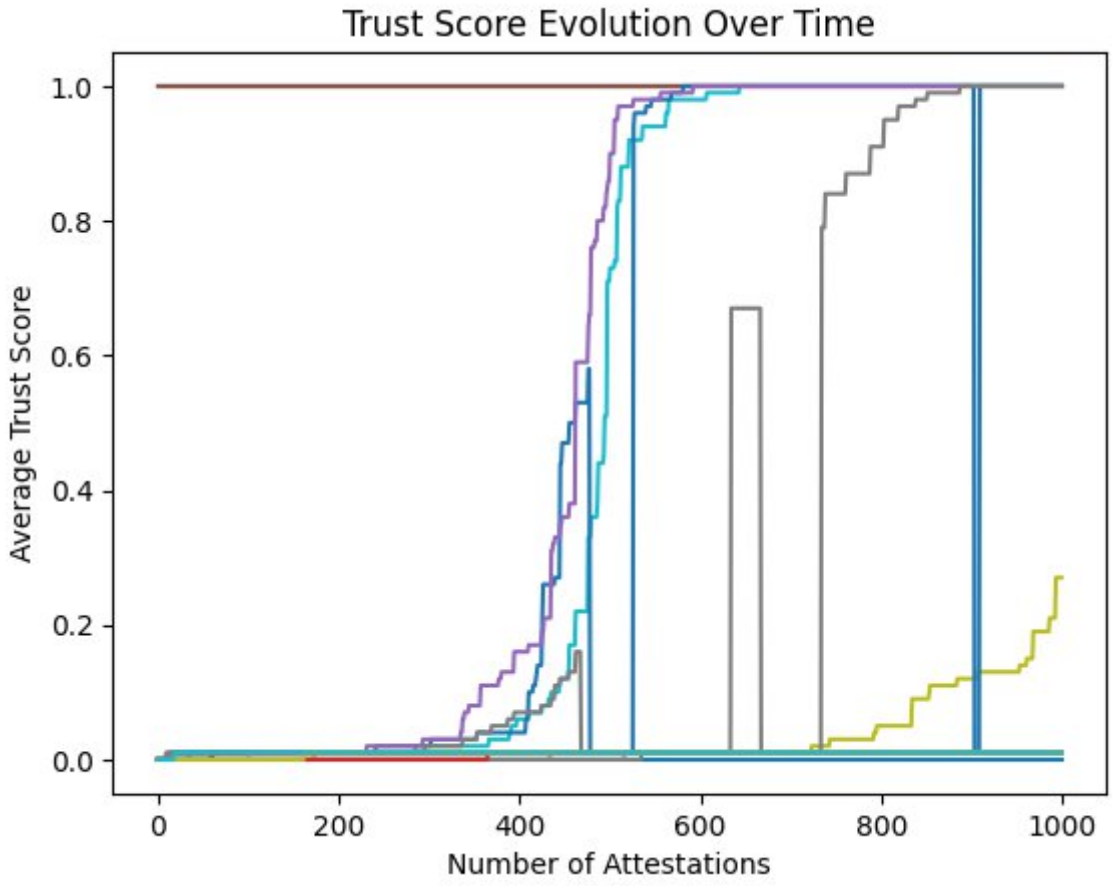

Figure 5: Trust score versus 'time' which is the number of attestations made over time

Model Robustness

Our simulations have shown that the model is robust against a variety of scenarios:

Dishonest Majority: Even if a majority of entities are dishonest, as long as there's a sufficient number of trusted entities, the model can still produce accurate trust scores.

Conflicting Attestations: In cases where there are conflicting attestations about the same claim, the model weighs them based on the trustworthiness of the attesters, ensuring that more credible claims have a higher impact.

Section 4: How it could work in practice

Bootstrapping with Trusted Sources

The initial phase of implementing our trust propagation model is crucial. The algorithm's effectiveness is heavily reliant on the initial set of trusted sources. These sources serve as the foundation upon which subsequent attestations are evaluated. Consider the bootstrapping of blockchain networks. There is always some trust in the initial set of miners and stakers, but over time, decentralization grows and trust is minimized.

Selection of Trusted Sources: The initial group should comprise entities that are widely recognized for their credibility and expertise in the domain. For instance, in the context of evaluating the safety of smart contracts, established auditors and educators in the blockchain space would be ideal candidates.

Avoiding Collusion: It's essential to ensure that the initial group of trusted sources is diverse and independent. This minimizes the risk of collusion, which could compromise the integrity of the resulting confidence calculations.

Public Attestations: Trusted sources would be encouraged to make public attestations about various claims. For instance, auditors could attest to the safety of a smart contract, while educators might vouch for the proficiency of their students. Note that attestations don’t have to be binary (True/False). If an auditor has only audited 50% of a smart contract, they can ‘stake’ 50% of their reputation on the confidence of the attestation, by applying at 50% weight. For liability reasons, we assume no auditor can be 100% confident. As it is currently, the maximum confidence in an attestation is 90%.

Taking these factors into account, we believe that onboarding auditing firms and DAOs would lead to a successful network on attestations. Organizations such as Code4rena or Immunefi would be a good starting point. **To repeat, our main request of these platforms is to make simple signed messages on EAS, attesting to the safety of the smart contracts they have audited.**

Growing the Network

Once the initial framework is in place, the network can be expanded organically:

Propagation of Trust: Trusted sources can further propagate their trust to other entities. For example, auditors might propagate confidence in their employees or contributors, while educators could do the same for their students.

Dynamic Confidence Scores: As more attestations are made, the confidence scores of various claims and entities will be continuously updated. This dynamic nature ensures that the system remains responsive to changes, such as contract upgrades, discovery of vulnerabilities, or shifts in public sentiment.

Re-evaluate Trusted Sources: Once the network has grown, we can remove the ‘training wheels’ of trusted sources, and recalculate the confidence in the original trusted sources.

Section 4b: Personalized Confidence Scores

While the primary goal is to provide a general confidence score based on collective wisdom, there's also value in allowing for personalization, and the method described in this paper is capable of calculating personalized confidence scores but utilizing:

Incorporating Private Attestations: EAS's capability to include non-public attestations means that users can factor in private claims or endorsements that might not be visible to the broader network.

Weighted Trust: Users might have their own set of trusted entities. By allowing them to assign greater weight to these sources, they can generate custom confidence scores that align more closely with their personal beliefs or biases.

Custom Algorithm Runs: Users can adjust parameters, include/exclude certain attestations, and run the algorithm to generate personalized confidence scores. This could be done on their own systems, in a private cloud, or even as a paid service provided by our platform.

Future work:

We would like to perform further research into the following topics:

Incorporate Time Decay: Older attestations might be given less weight compared to recent ones, ensuring the system remains responsive to current sentiments.

Mixing Claim Types: The current implementation assumes all claims are roughly the same type (e.g. Humanity, Smart Contract Safety, Creativity) because it we assume that humans attest to humans, and smart contract auditors confidence scores are not related to Creativity scores. If an entity has high trust in one area, it does not translate to high trust in a different type of claim. The solution here could be service providers like Gitcoin Passport that mix claims, or further adjustments to the algorithm presented here.

Bias and Echo Chambers

A potential unintended consequence of the system is the creation of echo chambers, where entities only trust attestations that align with their existing beliefs. To reduce risk of bias, we plan to employ the following practices:

Diversity of Trusted Sources: Ensuring a diverse set of initial trusted sources can help in presenting a more balanced view.

Transparency in Algorithm: Making the trust propagation algorithm transparent and open-source can help in identifying and rectifying biases.

References:

[1] Grüner, Andreas, et al. "A quantifiable trust model for blockchain-based identity management." *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018.

[2] EigenTrust for Web3: https://docs.karma3labs.com/introduction/our-thesis