zk-promises: Anonymous Moderation, Reputation, and Blocking from Anonymous Credentials with Callbacks

Maurice Shih [1] Michael Rosenberg [1] Hari Kailad [1] Ian Miers [1]

[1]University of Maryland

Introduction

- Anonymity is essential for free speech. Forums such as 4chan allow users to remain anonymous, but they suffer from one important problem: **moderation**.
- Anonymous platforms lack **accountability** and platforms with moderation lack **anonymity**.
- For example, Wikipedia bans users if they make malicious edits – how will the server know who to ban if the user is anonymous?

Background

- **Anonymous blocklisting** systems allow users to remain anonymous while having a single bit of state (banned or not banned). For example, systems such as SNARKBlock use zero-knowledge proofs to force users to remain banned.
- **Anonymous reputation** systems allow users to have arbitrary state such as reputation or karma. Prior systems support arbitrary state, but are not **asynchronous**.
- Current anonymous reputation systems are limited in their abilities or are not efficient enough to be used in practice.

Preliminaries

- We say $c = \mathrm{Com}(x; r)$ to some data x is a commitment if c is hiding (hides what x is), and is binding (can't find different x with commitment c).
- Let $f(x, w) \to \{0, 1\}$ be a function. Then a zero-knowledge proof system allows one to produce a proof π_f, which asserts that the user knows w such that $f(x, w) = 1$.

Objective

Can we construct an anonymous reputation system where

- **Anonymous**: users remain fully anonymous
- **Complex feedback**: updates to user state are Turing complete
- **Asynchronous**: updates to user state can occur while the user is offline

which is also **practically efficient** for people with standard devices (such as a laptop or smartphone)?

System

- Our system is built using zk-objects. Let $\mathcal{U}_i$ contain some user state, for example $\mathcal{U}_i = \{\text{reputation}, \text{last\_post\_time}\}$.
- The server stores a list $\mathcal{L}$ of $C_i = \mathrm{Com}(\mathcal{U}_i; r_i)$. A user can then prove statements about their object.

$$f(\mathcal{L}, \mathcal{U}_i) = \mathrm{Com}(\mathcal{U}_i; r_i) \in \mathcal{L} \wedge (\text{any statement})$$

- Serial numbers are used for updating a user following the ZCash model to prevent double-spending:

$$\pi : \mathrm{Com}(\mathcal{U}_i; r_i) \in \mathcal{L} \wedge \Phi(\mathcal{U}_i, \mathcal{U}_i') \wedge \mathcal{U}_i.\text{serial} = s \wedge \mathrm{Com}(\mathcal{U}_i'; r_i') = C_i'$$

Adding anonymous feedback

- On an update (post to forum such as Reddit), **user hands the server a one-time ticket**

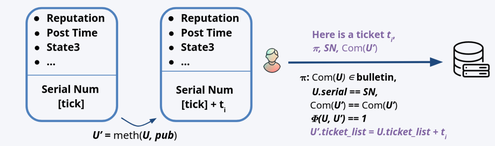

Figure 1. An update with tickets (callbacks).

- This **ticket** or **callback** may be "called" later by posting it online on a bulletin board $\mathcal{L}_c$
- User is **forced to update state** by producing a zero knowledge proof that the update occurred if $t_i \in \mathcal{L}_c$

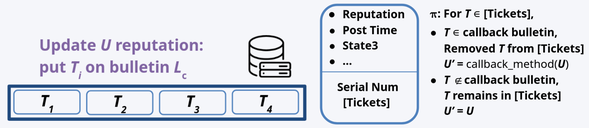

Figure 2. Server calling and user settling tickets.

Future

- **Private state:** Users can currently see their own state
- **Folding:** Folding techniques in proof systems can make settling tickets faster

Resources

Read the paper!

Results

Performance Microbenchmarks

- **Making posts:** 328ms proving time, scales logarithmically with the maximum number of callbacks.
- **Settling one callback:** 512ms proving time, scales logarithmically.
- Constant time if server is not decentralized.

Further Optimizations and Functionality

- **Rate limiting:** Users may be rate limited via a leaky bucket algorithm.
- **Batched scanning:** Settle multiple tickets at once, faster proving time (see below).
- **Expiry:** Ticket list doesn't grow forever, tickets expire.
- **Methods and arguments:** Different tickets can have different methods, and can call tickets with (encrypted) arguments
- **High dimensional state:** Multiple service providers may access different parts of a vector of state.

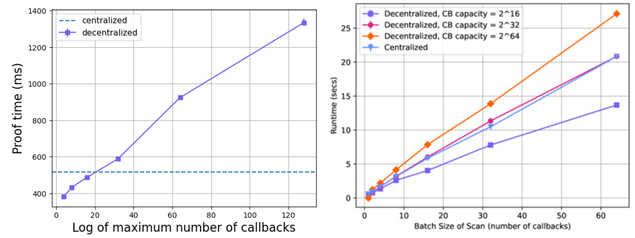

Figure 3. Proving time for settling a single outstanding ticket, and proving time when batching n outstanding tickets.

Conclusion

- Built first **efficient** and **versatile** anonymous reputation system, which supports Turing complete methods and advanced functionality, using cryptography and zero-knowledge proofs.
- Fully **programmable**, **asynchronous**, and **scalable**.
- Can be used for anonymous forums, oblivious VPNs, whistleblowing, and cryptocurrency reputation.