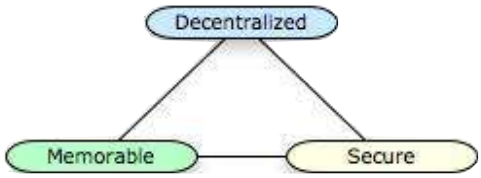
A worse-is-better approach to solving Zooko's Triangle

Victor Grey

3 February 2005

Zooko's Triangle (http://zooko.com/distnames.html) asserts that online names can be "decentralized, secure, human-meaningful: choose two."

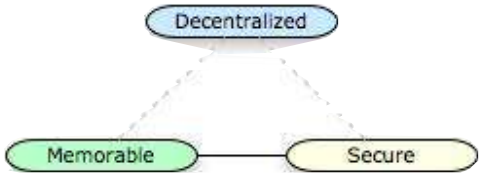


To be truly secure, Zooko says, a name must be self-authenticating, e.g. a name that contains or is a public key or PK fingerprint can be dereferenced to data that is signed with the corresponding private key.

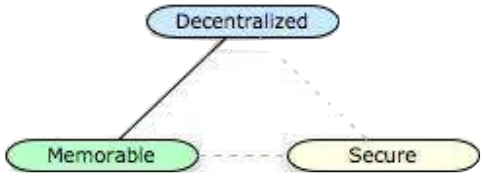
To be decentralized, a naming system must not depend on a centralized authority (e.g DNS).

To be human-meaningful, a name must be able to be remembered by the owner, and easily communicated, for example printed on a business card or verbally. I'll call this quality "memorable".

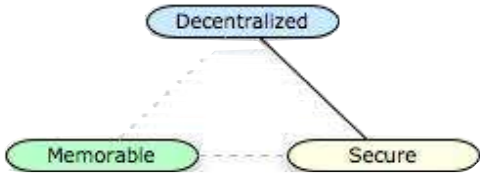
You can have names that are (paraphrasing Zooko again) secure and memorable but depend on a central authority to resolve.



You can have names that are decentralized and memorable, but are not secure, "in the sense that anyone can change any name to point to any value at any time."



And, you can have names that are decentralized and secure, using cryptographic keys that are not memorable.



Zooko suggests a solution, in that you can have decentralized-secure names that are translated to a memorable pointer by a computer agent that runs locally to the name owner and is loyal to them and under their control.

But how can such a system achieve widespread adoption? There are many examples of systems that failed by requiring users to download or otherwise acquire software, and install it and learn how to use it. The inertia of the computer using public is vast. Users will generally endure significant pain rather than climb the installation and learning curve of a new system.

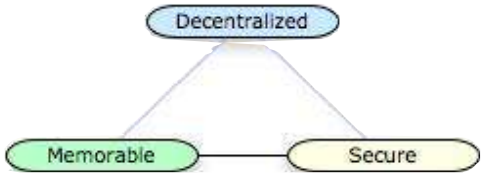
2idi is adopting the worse-is-better approach to this issue. We take as our example the design of the most widely adopted system the planet has ever seen, the world wide web. To quote Clay Shirky (http://www.shirky.com/writings/evolve.html):

> "HTTP and HTML are the Whoopee Cushion and Joy Buzzer of Internet protocols, only comprehensible as elaborate practical jokes. For anyone who has tried to accomplish anything serious on the Web, it's pretty obvious that of the various implementations of a worldwide hypertext protocol, we have the worst one possible.
>
> Except, of course, for all the others."

It was adopted however, because it was extremely easy for developers to grasp and start using, it was good enough, and it was highly evolvable. Shirky again:

> "Centrally designed protocols start out strong and improve logarithmically. Evolvable protocols start out weak and improve exponentially. It's dinosaurs vs. mammals, and the mammals win every time. The Web is not the perfect hypertext protocol, just the best one that's also currently practical. Infrastructure built on evolvable protocols will always be partially incomplete, partially wrong and ultimately better designed than its competition."

We are starting with a system that is somewhat centralized, based on open source and open protocols, and, this is the important point, just barely good enough to work without requiring users to make any great leaps right away.



It is our intention that it evolve in the direction of decentralization, by pushing I-brokers and registries closer to the user, until the solution that Zooko himself suggested, local agents, can finally be nearly universally adopted.

Our challenge is to design our systems to be as minimalist as possible while still highly valuable to users, and to not include anything in our design which would prevent the desired evolution. To this end, an important problem will be to describe a way to merge independent namespaces without collisions, and without the need to appeal to any particular higher authority (or only to any mutually agreed and perhaps mutually created higher authority).

Appendix: Resolving Conflicts Between Independent Roots[1]

The question has been asked, "What if two or more independent communities ignore the above advice to use a globally unique cross-reference as their root and instead choose conflicting relative identifiers to represent their root? How could these communities resolve that conflict and begin to safely cross-resolve XRIs?"

An example would be two communities who both choose the relative i-number `!1000` as their root XRI. Under this root i-number, all other i-numbers would be unique, but because they both use the same root i-number, there could be an infinite number of collisions. For example, there would be no way to distinguish the following two XRIs that are identical in both communities:

```
Community A    xri://!1000/!1234
Community B    xri://!1000/!1234
```

The solution is for the two communities to federate by each establishing a unique i-number relative to another shared root, then cross-referencing their existing community i-numbers in this new shared context.

For example, if Community A and Community B agreed to use the new shared context root `!1111`, and establish that relative to this new root Community A would be `!1111*!A` and Community B would be `!1111*!B`, both communities can now reference all their existing XRIs in this new shared context without any collisions. For example:

```
Community A    xri://!1111*!A/!1000/!1234
Community B    xri://!1111*!B/!1000/!1234
```

Note that in this example, because the existing root i-numbers were relative, they could be merged directly below the new shared root. Had the two communities used a global i-number for their roots instead (for example, if they had both declared that their root was the global i-number `!!1000`), then decided to federate, XRI syntax would require the current i-numbers to appear as cross-references, as shown below:

```
Community A    xri://!1111*!A/(!!1000/!1234)
Community B    xri://!1111*!B/(!!1000/!1234)
```

What changes would have to be made to the registries and/or people's address books?

Each community root registry would need to add an entry recognizing the new shared root and its own identifier under the shared root. (In XDI universal graph terms, this would mean adding a synonym to the current logical root resource node and a backref to the new shared root node.)

Updating address books in each community could be an organic process. The first phase would not be to update the address books themselves but rather the community resolvers used by the address books. Each community resolver would continue referencing its own default root for unqualified addresses (addresses that do not use the new shared context), but would begin recognizing qualified addresses in the other community.

The second phase would be for the address books to be updated by adding XRI synonyms for every unqualified community entry. This would allow the same resource to be recognized both as an unqualified and a qualified community address. The community could slowly migrate over to using qualified addresses (like migrating from 5-digit to 9-digit zip codes) until the address books were updated.

---

[1] From section 3 of http://xrixdi.idcommons.net/moin.cgi/XriPeerToPeerAddressing