Names: Distributed, Secure, Human-Readable: Choose Two

version 0.9.1, 2001-10-12

Please do not propagate this information widely yet. I'm still working on it.

Non-Self-Authenticating Names Cannot Span Trust Boundaries

There are two kinds of name-value pairs: those that are self-authenticating and those that aren't. A name-value pair is "self-authenticating" if, given the name-value pair, you can verify on your own that the mapping from that name to that value is correct. (For example, if the name is the secure hash of the value, then it is easy to verify that the value you have is the one and only valid value that the name maps to.)

If your distributed namespace confines itself to serving self-authenticating name-value pairs, then you can easily be free from the risk of deception and the only remaining security issue is to prevent denial of service attacks.

If your distributed namespace allows non-self-authenticating name-value pairs, then I doubt that it will be secure. I have not seen any proof that such a system is even possible in principle, it isn't even clear what the intended behavior *should* be, and the best designs that I have seen are complex, limited, and risky.

Self-Authenticating Names Cannot Be Memorized By Humans

Two examples of self-authenticating name-value pairs are those where the name is the secure hash of the value (Freenet→'s "CHKs" and Mojo Nation→'s *mojoids*) and those where the name includes the id of a public key and the value includes a digital signature from the corresponding private key (Freenet's "SVKs" and the Self-Certifying File System→'s names).

Okay, so if self-authenticating pairs can be securely distributed and non-self-authenticating pairs cannot, then why don't we just satisfy ourselves with the former? Because they are not human-readable.

> "Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design out protocols around their limitations.)"
>
> -- *Kaufman, Perlman, and Speciner quoted in Ross Anderson's "Security Engineering"*

But Your Computer Can Remember Things For You

Okay, that's funny, but we *don't* have to design *every* namespace to support human-usable keys.

If you are designing a distributed namespace, think about whether you *have* to support human-usable, non-self-authenticating keys, at the cost of losing the ability to safely span trust boundaries. For some applications, you do need human-usable keys, for others you don't.

And then after you've thought about that, think about the fact that your human user might have a loyal computer assistant at hand, which can translate self-authenticating names into human-usable names (Pet Names Markup Language→). If so, then this moves your application from the "requires human-usable keys" category to the "does not require human-usable keys" category.

Two Ways To Design A Distributed Namespace That Does Not Span Trust Boundaries

The current state of the art as far as I have seen tends to fall into one of two traps.

- First trap:
  - Assume that you need arbitrary keys including non-self-authenticating ones.
  - Don't think about trust boundaries.
  - Publish.
  - Try to figure out how to make it secure without introducing a centralized "trusted third party".
  - Go to step 4.
- Second trap:
  - Assume that you need arbitrary keys.
  - Think about trust boundaries, and solve by delegating to a "trusted third party".
  - Pay a tax to MicrosoftNSIVerisignICANNUSGovInc. for every packet.

You Can't Have It All, But You Can Have Some Of It

To summarize, you cannot have a namespace which has all three of: distributed (in the sense that there is no central authority which can control the namespace, which is the same as saying that the namespace spans trust boundaries), secure (in the sense that name lookups cannot be forced to return incorrect values by an attacker, where the definition of "incorrect" is determined by some universal policy of name ownership), and having human-usable keys.

So what should you do? There are at least four alternatives:

1. You can have a namespace that is secure and allows human-usable keys, but that is centralized in terms of trust -- all participants must trust a central authority (or perhaps a hierarchy of authorities with each authority trusting a more central authority, and with one ultimate authority at the top). Many people seem to think that this is the only possible kind of namespace (perhaps because of their familiarity with DNS, which is an example of this design), and corporations like VeriSign/NSI, Microsoft, Sun, and others are working hard to gain control of the top spot, which they think to be a natural monopoly rich in unexploited profits.
2. You can have a namespace that is distributed and uses human-readable keys but is not secure, in the sense that anyone can change any name to point to any value at any time. Freenet has a namespace like this, but the Freenet developers encourage people to use CHKs or SVKs instead.
3. You can have a namespace that is distributed and secure but that does not allow human-usable keys. Freenet's CHKs, Mojo Nation's *mojoids*, and SPKI→ certificates do this for immutable key-value pairs (where a given name refers to a bitwise copy of an object and that name can never be made to refer to any other bit pattern) and Freenet's SVKs and Self-Certifying File System's directory names do this for mutable key-value pairs (where a given name, which includes a public key, can be made to refer to a new object in addition to the old object, but only by using the private key).
4. You can have a namespace that is distributed and secure, as in #3, and in addition you can offer your users a computer agent, which runs locally to them and is loyal to them, which translates the self-authenticating names into human readable names for their use. The Pet Names Markup Language→ is a specification for how such an agent would interface with the user and with the secure distributed namespace. Since most uses of a distributed namespace will already include a user agent (e.g. a web browser, handheld wireless device, an e-mail user agent, etc.), this requirement is not as burdensome as it might appear. It prevents humans from speaking the names to each other vocally (but allows them to beam them to each other with handheld or worn devices), and it prevents you from using the names in advertisements on billboards, radio and television. This last limitation may be seen as a feature.
5. Show me that I'm wrong. I didn't *prove* that it is impossible to have all three features, I only said that I doubted that your namespace will have all three. If you think your namespace design can provide all three, then lay out an argument that it can do so. The first step will be to specify what it *means* actually for the distributed human-usable names to be secure, which is the same as specifying what universal policy should govern ownership of names. (For example, in the case of CHKs, to be secure means that you can't have a collision such that one CHK identifies two different bitstrings which, conveniently enough, is part of the definition of security for cryptographic hashes. In the case of SVKs, to be secure means that only the holder of the private key can change the mapping from a given SVK to its object, which is conveniently similar to the traditional notion of security for digital signatures.) If your distributed namespace satisfies all three criteria then (a) you should carefully justify your claim that it does so, starting with a specification of what the ownership policy is, and (b) it will be the first published design that does so!

Thanks to Mark Miller→ for teaching me about the important distinction between self-authenticating and non-self-authenticating key-value pairs and for helpful comments on an earlier draft of this article.

Zooko
Last modified: Sun Oct 14 18:09:33 PDT 2001