Opaque Identifier Aliases*

Daniel Hardman

2026-07-09

Abstract

Cryptographic identifiers — the keys, DIDs, and autonomic identifiers that anchor digital identity — are strings no human can remember, compare, or choose between. This primer, a companion to Zooko and Houdini and to the COIA naming convention, is about the humble tool that makes them livable: the alias, a friendly local label a person attaches to an opaque string. It argues that an alias is best understood as a private nickname — non-unique, changeable, meaningful only to its creator, and never to be mistaken for the identifier itself — and it examines the properties that follow, the reason a shared naming convention beats ad-hoc labels, and the single most dangerous mistake in the whole design: confusing what you have *proved* about the party behind an identifier with what you have merely *guessed*. It closes by showing how good aliases let software choose the right identifier for a person most of the time without asking, ask a sensible question when it cannot, and connect people to one another one facet at a time.

A friendly name is a convenience, not a fact — and confusing the two is where trust goes to die.

> "What's in a name? That which we call a rose / By any other name would smell as sweet." — William Shakespeare, *Romeo and Juliet*, Act II, Scene 2

The promise of a different essay

Zooko and Houdini [1] — a parable about *Zooko's Triangle*, the old observation that a name can be at most two of memorable, global, and secure — ended with a promise. Having argued that a friendly name lives in the eye of the beholder — that the same person is "Mom" to one contact and "Regina Q. Public" to another, with no conflict — it noted that a convention for choosing such names "is the subject of a different essay". This is a companion to that different essay. Where the convention itself — COIA [2] — is a precise recipe, this piece is about the *why* underneath it, and about a few things a recipe cannot say.

Start with the problem the friendly name solves. A modern digital identity is anchored by an **opaque identifier** — a public key, a decentralized identifier (DID), a KERI autonomic identifier (AID), a payment address. It looks like this:

```
did:example:EMyYnLzlJDJskqojipIMivAKHWeZofhWiYjB79uszynS
```

A computer loves this string. A human cannot remember it, cannot read it aloud without errors, cannot tell it apart from a near-twin at a glance, and cannot choose the right one from a list of forty.

*daniel.hardman@gmail.com

And there *will* be a list of forty. As Identity Facets [3] argues, a person acts through many facets — employee, officer, parent, patient, citizen — and good practice gives each facet its own identifier, so that trust and security never bleed across the boundaries between them. Solve identity well and you have multiplied the opaque strings. The friendly name is what keeps that multiplication from crushing the user.

An alias is a private nickname

The friendly name has a technical name of its own: an **alias** (in the older literature, a *petname*). And the single most useful thing to understand about an alias is what kind of thing it is *not*.

An alias is not an identifier. It is a **nickname** — the label you, privately, hang on an identifier so that you can find it again. Think of the contact list in your phone. You have an entry called "Mom". It is perfect for you and useless to anyone else: your friend's phone has a different "Mom", and neither of you is ever confused, because you each look up "Mom" only in your own list. Four properties come with it, and every one of them is load-bearing.

- **An alias is local.** It belongs to the person who created it and means something only in that person's world. It is not published, not shared, not a public handle. A stranger might overhear it, but it was never built to be reshared.
- **An alias is not unique.** Two people can label the same identifier differently, and one person can use the same label — "the accountant" — for two different identifiers over the years. Uniqueness is the job of the identifier underneath, never of the name on top.
- **An alias can change.** You may rename a contact tomorrow on a whim. The identifier it points to does not change when you do; only your label moves.
- **An alias must never be parsed for meaning by anyone but its creator.** This is the rule that, broken, plants the subtlest bugs. If my software reads *your* alias and concludes something about the party behind it — treats the string as data — it has trusted a private note it had no business trusting. The name is a memory aid, not evidence.

Miss the category and you write broken logic: treat an alias as an identifier and you get collisions; cache it as though it were stable and you get staleness; parse a stranger's alias for meaning and you get an attack surface. The rose is the identifier. The alias is only what we happen to call it today.

A convention, not a free-for-all

If names are private and personal, why standardize them at all? Because a good convention is a teacher.

Left to invent their own labels, people produce a mess — inconsistent, half-forgotten schemes that drift over time and confuse their own authors a year later. Worse, a naïve labeller never learns the distinctions that keep them safe. A convention fixes both problems, not by dictating, but by example. Give a person a handful of well-formed examples and they tend to absorb, without being lectured, that the identity they are labelling has a *who*, a *role*, and a *scope* — and that these are the things that matter when deciding which identifier to use.

Those three questions — who is this, acting as what, in what context — are the heart of the COIA convention, and are treated fully there [2]; I won't restate the algorithm. The point worth making here is conceptual: the same three answers that produce a good *name* also capture the *meaning* a

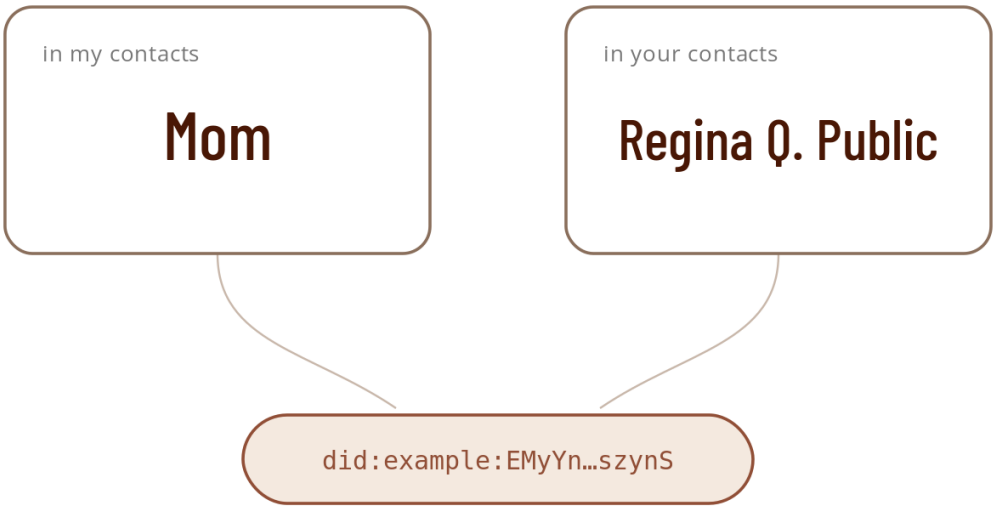

Figure 1: Two friendly names, one identifier underneath: each contact list labels the same identifier for its own use.

system must track anyway. A label is for a human, in one language, at a glance; the structured who/role/scope behind it is for software, for other languages, and for the day the label proves too thin. Name and metadata are two expressions of one underlying fact.

Proved versus guessed

Now the sharpest edge in the whole design, and the reason aliases are a security topic and not merely a usability one.

When your system shows you an alias for a *remote* party — "Cecilia as CEO at Acme" — that name asserts something: that the identifier belongs to Cecilia, who is Acme's CEO. But how does your system *know* that? There are two very different answers, and they must never be confused.

Either you have **proved** it — confirmed, over a trusted channel, that the identifier really is controlled by the party you think — or you have merely **guessed** it, accepting a claim that arrived with the identifier and has not been checked. The gap between proved and guessed is the gap a **man-in-the-middle** attacker lives in — someone who secretly interposes himself between you and the party you mean to reach. He hands you an identifier and a plausible name; if your software renders that guess with the same confidence it renders a proof, it has invited you to trust an impostor.

A good convention refuses to let the two blur. COIA reserves a flag — a leading `0` — that means exactly *"unconfirmed: there may be a man in the middle here"*, and it insists that any alias for a remote party wear that flag until the doubt is deliberately cleared — so an unproved contact reads `0cecilia-as-ceo-at-acme`, the leading `0` a standing caveat, and only once the doubt is cleared does the flag drop to leave a plain `cecilia-as-ceo-at-acme`. The name still helps you find the

contact; it just stops short of vouching for them.

And how is the doubt cleared? By comparison — a problem of its own, because it asks a human to confirm that two copies of an opaque identifier are the same. People are terrible at comparing long strings character by character [4], so the honest way to clear the flag is to change the task from reading to *recognizing*: render the identifier as a picture that a substituted key would visibly disturb. That is the job of entviz [4], a scheme that turns an identifier into a small distinctive image so that a substituted key produces a visibly different picture; its perceptual limits have been estimated [5]. The pipeline is clean: an unproved alias carries the `0` flag; the parties compare their identifiers over a trusted channel — a video call where they can see each other, an in-person QR scan — by eye; and only then does the flag come off. Naming raises the doubt honestly; visualization resolves it.

Choosing for the user

Put the pieces together and software can pick the right identifier for a person *without asking them a thing* — most of the time.

The reason is that the system usually knows the context. When a workflow requires an officer of a company to sign, the system knows which facet — and therefore which identifier and which alias — the moment calls for. It need not interrogate the user about cryptography; it simply uses the right key. And because acts that are genuinely ambiguous in capacity are the exception, this covers the great majority of interactions.

But not all. Some acts are genuinely valid in more than one capacity. Cecilia can sign a loan as a private citizen *or* as her company's officer, and the two have different consequences (a point Identity Facets [3] develops). Here the system cannot and must not guess. The saving grace is that, precisely because the ambiguity is real, the question is one a human can answer with ease: *"Are you signing this as yourself, or as CEO of Acme?"* That is not a cryptographic riddle; it is a question about the person's own life, and the aliases make the options legible. Ask it only when it is real, phrase it in the user's terms, and the rare interruption reassures rather than confuses.

Connections, and the line between introducing and proving

One last consequence. Because each facet has its own identifier and its own alias, we do not connect people to *people*; we connect them to *facets*. When a reporter interviews Cecilia in her role as CEO, she meets Cecilia-the-CEO's identifier, and nothing else — not Cecilia the patient, not Cecilia the parent. The boundary is preserved by construction.

This lets software be generous with a chore it has no reason to hoard: **introductions**. It can automatically introduce colleagues who plainly need to collaborate — hand them one another's facet-appropriate identifiers and aliases — sparing everyone the busywork of maintaining contact lists by hand.

But introducing is not the same as proving, and telling the two apart is the whole discipline of this essay in miniature. To *introduce* is to say "here is an identifier you'll probably want". To *prove* is to close the gap between guessed and proved from earlier — to assemble the evidence that no man in the middle stands behind it: a challenge answered over a trusted channel, a high-assurance credential produced on demand, two copies of an identifier rendered and found identical. Much of that assembly a machine can and should perform; gathering evidence is mechanical work, and

the less of it we put on the user, the better. What a machine must not do is take the last step — the moment a human weighs that evidence and decides it is good enough to trust *this* party, for *this* purpose, now. That step is the *verification* of the proof, and it is the part that has to stay human. Automate the introduction. Automate the proof where you can. But never automate the verification of the proof.

What the machinery buys

Done well, none of this shows. The user sees a friendly name at the right moment, is asked a sensible question on the rare occasion it is needed, and is spared the opaque strings entirely. Underneath, two questions have been answered by two different tools — and they are the spine of humane cryptographic identity. *Which identifier is this?* is answered by the alias — a private nickname, governed by a convention, honest about what it has proved. *Is this the same identifier?* is answered by comparison — a picture a human can check. Naming for the many facets; comparison for the sameness beneath them. Neither alone is enough; together they turn a fistful of unreadable strings into something a person can actually live with.

References

[1] Hardman, D. 2024. Zooko and Houdini: A Flatland Parable. Codecraft Papers. Retrieved from https://dhh1128.github.io/papers/zh.html

[2] Hardman, D. 2025. Conventions for Opaque Identifier Aliases (COIA). Retrieved from https://dhh1128.github.io/coia

[3] Hardman, D. 2026. Identity Facets. Codecraft Papers. Retrieved from https://dhh1128.github.io/papers/if.html

[4] Hardman, D. 2026. Amplifying Difference: Perceptual Design and Verification of Human-Centric Entropy Visualizations. Codecraft Papers. Retrieved from https://dhh1128.github.io/papers/amp-diff.html (SSRN DOI: https://doi.org/10.2139/ssrn.6979798)

[5] Hardman, D. 2026. Measuring the Glance: An Adversarial Estimate of Habituated Perceptual Entropy in entviz and SSH Randomart. Codecraft Papers. Retrieved from https://dhh1128.github.io/papers/m-glance.html (SSRN DOI: https://doi.org/10.2139/ssrn.6979878)