

Identity Agents

Paul Trevithick, The Mee Foundation

March 9, 2023

1 Power Asymmetry

While the internet has brought an explosion of new services to billions of users, there is room for improvement. Specifically, we focus on the power asymmetry between users and the digital service providers regarding the user's digital identity and personal data. People lack control (i.e. power) over what they refer to as their *own* data.

In order to reinforce that we're talking about power relations, we use the term *computational power* to refer to software that works (i.e. provides agency) "on the user's side" for, and *exclusively* on behalf of, the user.

1.1 Autonomy

Independence. Unlike the real world, we have no independent identity online. It could be said that we have no *digital embodiment*¹. The online identities we do have are provided to us by digital service providers (e.g. a Facebook identity, or an Amazon account) and thus we don't have *self-sovereign identities* [add reference to Drummond's book]

Portability. Providers have our data and as a result, we are held hostage. As Tim Berners-Lee has recently said, "We are locked into platforms that hold our data" [need to check and get reference]. Data portability remains more of an aspiration than a reality.

Trust. In the offline world you can autonomously present you drivers license to a wine seller to prove you exceed the minimum drinking age. The wine seller trusts the issuer of the license, not you. There is no standard way to do the same thing online. Further, the interaction is privacy respecting because the presentation event is never learned of by the issuer. Although there are domain-specific exceptions (e.g. cryptocurrency), online there is no common, generalized way for you to prove something about yourself as stated by one party about you, to another party.

¹P. Windley, Personal communication, September 2022

Delegation. In the offline world one entity can grant access to some resource to another entity. For example, I could give my car keys to a friend so they could borrow my car. There is no standard, or secure way to do this online. This is especially problematic in healthcare scenarios where a caregiver needs to gain access to electronic health-related data about another person.

1.2 First-parties

When we create an account on a provider's app/site, we are creating a relationship with the provider as a first-party. There are several power asymmetries in this relationship.

Provider-defined Privacy Policies. Users of a provider's site/app are given the option to review the privacy policies dictated by the provider. The policy's purpose is mostly to protect the provider's interests. The extent to which it protects the user's interest is determined by the regulatory regime they are under, and their degree of adherence to it. In practice the user doesn't have the time to read an average of 100+ policies for the providers they use because they lack computational power on their side to make sense of them.

User-defined Privacy Policies. With the notable exception of the Global Privacy Control, users generally lack the computational power

User Rights. More problematic still, is that even when the privacy laws state that the user has rights, for example to access, correct and delete their data the user lacks computational power on their side to exercise these rights. Today the user has to send written requests to get their data, request that it be updated or deleted, etc. None of this is practical because it is too burdensome on the user. Without this power on the users side, users live in doubt and fear as to whether providers will use their data against their interests, share their data with third-parties.

Inferences. [page 66 of Zucked "Eli Pariser need to give users control of their bubble (interests profile)]

1.3 Third-parties and surveillance

Third-party tracking. Whereas the user is at least aware when they sign up at a first-party site/app that their on-site interactions are known to the provider of that site/app, there are hundreds of third-parties that the user isn't even aware of that track the user, and assemble databases about us. These databases create privacy risks and vulnerabilities. We have little transparency into what's being gathered, where it's being shared and how it's being used. Much of this third-party tracking is enabled by the first-parties for economic reasons (e.g. first-parties placing dozens of third-party tracking cookies on the user's browser).

Surveillance-based targeted advertising. In general, targeted advertising involves four steps:

(1) collection of observations about the user by a first- or third-party, (2) synthesis of an "ad profile" from these observations, (3) matching this ad profile against available "target audiences" (i.e. characteristics of whom the advertiser wishes to reach, advertising budget, etc.) from advertisers through a bidding process, and (4) displaying the winning ad. Surveillance-based targeted advertising when step (1) is achieved through third-party tracking as described above. The third-party vendors have the power to perform the tracking and synthesis of a user's "ad profile", and since the user lacks this computational power on their side there is no alternative. Users have no say in their own ad profiles—never seeing them and never with the ability to correct them.

Data brokers. Data brokers who buy and sell personal data to other brokers, to advertisers, adtech firms and first-party publishers provide liquidity (along with a host of privacy threats) in the personal data shadow marketplace because user's lack the computational power to provide data about themselves.

1.4 Convenience

The prevailing architecture of the internet wherein each provider manages their own separate "silo" of information about you (i.e. your account) not only has all the challenges just mentioned. But the lack of computational power on the user's side also creates several inconveniences for the user.

Repetition. Endless repetition as you fill in forms

Password management. The average user has over 100 site and 25 apps in daily use. Managing passwords at each is realistically impossible without an automated password manager (computational power).

Account Management. The user has the inconvenient burden of maintaining their account information at 100+ silos. Even just updating contact or credit card information at dozens of sites without computational power on the user's side, this is tedious and time-consuming and encourages the user to spend more time at a smaller number of larger mega-sites.

2 Design Considerations

Our vision is to develop an identity agent to address the problems outlined in the previous section. This agent can represent the user and promote their interests online. In this section, we discuss a number of design considerations for this agent illustrating the each with use cases.

2.1 Decentralization

Many of the challenges described have their origin in an architecture that is provider-centric, not user-centric. In the provider centric model, each provider's app/site holds its own copy of the user's data. Each provider cares only about the narrow slice of the user that they see through their direct interactions. But to the user the situation is reversed. They sit at the center of 100+ connections with apps/sites radiating out from them. The user, for example, has a first name of Paul. It would be more convenient if Paul could type these four characters in once and have an agent communicate that information to the app/site when needed. But today each site asks "what is your first name?" and the user answers 100 times. What's needed is a decentralized architecture where each user has a personal datastore that they own and control.

2.2 Edge-centered vs. cloud-centered

Given that we need a per-user, user-centric decentralized architecture, where should this datastore live? We look at two options edge-centered and cloud-centered. Edge-centered means that the primary location for a user's personal datastore is on their own phones, and laptops and perhaps home servers. Cloud-centered means that the user's personal datastore is primarily held in the cloud (e.g. on a SOLID pod[add reference]). We say primary because there are usually use-cases that involved replicating/syncing some of the data to the "other" architecture.

Security. Although some may disagree, it is our contention that having a personal datastore on a personal device is more secure than in the cloud. Even if each platform alternative where equivalently secure, a cloud-centered architecture aggregates millions of personal datastores at one service provider and thereby creates millions of times the economic incentive for hackers to invest in attacking it.

Equity. The hosting costs of a cloud-centered solution must be paid for by some entity whereas user's typically own their edge devices and they are thus on a marginal basis "free". By equity we mean here that we need a solution that can be afforded by all socio-economic classes, and a solution that requires monthly hosting fees can thereby be ruled out.

Backup. One serious drawback of what's called a *non-custodial* edge-centered architecture (when compared to cloud-centered) is the need for the user's data to be backed up. This is not a problem if the user backs up their devices (e.g. to a cloud backup service), but many can't be relied on to be disciplined about this. [stats on phone vs. laptop backup]. [we assume the user has N_i1 device and that the agent is installed on N_i1 and that the agents replicate/sync]

2.3 Replication

If we assume an edge-centered design, we must solve the roaming problem. That is, we must support use cases where the user has more than one device and needs to be able to pick any one of them and have their person datastore be consistent across these devices (at least eventually). This requires that the agents of a given user implement replication and syncing. [discuss the (unfortunate and costly) need for relays to achieve P2P sync in some use cases].

2.4 Loyalty

Much of the power asymmetry described in the first section is due to economic incentives for providers to do just enough in the user's interest to keep them as a user or customer, but not more. Personal data, after all, is considered by business to be an asset class; the more of it that is collected and monetized the better. To have an agent that works not partially but exclusively on behalf of the user, the agent provider must not have an economic incentive to provide anything less but complete loyalty to the user's interests. Although there are other potential solutions (e.g. data cooperatives and data unions) we think the simplest approach is that the agent developer be a nonprofit organization that has no economic incentive to be anything but loyal to the user and to have no economic interest in their data. In fact there's no reason that that the agent developer to access or store the user's data.

2.5 Separation of app from db

[Talk about this from the point of view of the SOLID project or go-peer?? or ??]. Also talk about how this is inevitable. Desktop OSes had it, then in 2007?? iPhone got it, now in 2023! webapps will get it. Talk about automatic data portability]

2.6 Delegation

[Talk about delegated medical records use case]. [Talk about the Gropper Principle]

2.7 Multi-contextual

Zuckerberg has said "Having two identities for yourself is an example of a lack of integrity" [1]. He could not be more wrong. Let's take a step back and look at selfness and whoness.

2.7.1 Selfness and Whoness

In his last public speech² Kim Cameron³ introduced two useful definitions based on archaic English:

- **Selfness:** The sameness of a person or thing at all times or in all circumstances. The condition of being a single individual. The fact that a person or thing is itself and not something else. Individuality, personality.
- **Whoness:** Who or what a person or thing is. A distinct impression of a single person or thing presented to or perceived by others. A set of characteristics or a description that distinguishes a person or thing from others.

The following diagram illustrates these concepts and introduces the notion of context:



2.8 Open source

In order to trust that the agent does what we claim, we need transparency which open source can provide. Further this is an ambitious project and we need to nurture the creation of a community of developers help build it.

²www.youtube.com/watch?v=9DExNTY3QAk

³[en.wikipedia.org/wiki/Kim_Cameron_\(computer_scientist\)](http://en.wikipedia.org/wiki/Kim_Cameron_(computer_scientist))

2.9 Data Governance

Once data is shared from the agent to a first-party there are no technical means to constraint what the recipient does with it. No technical means can prevent them from selling it others, for example. Instead, legal means must be employed. Rather than wait for privacy regulations to get strong enough, we propose that first-parties sign a Human Information License to license the user’s information under terms that are fair and balanced and respect the user’s privacy rights. The contract can be signed by an entity that represents the user to make it effortless for the user.

2.10 User rights

[Talk about how even the best privacy legislation is impotent in practice to protect users because they don’t require associated technical means to implement them]

2.11 Enforcement

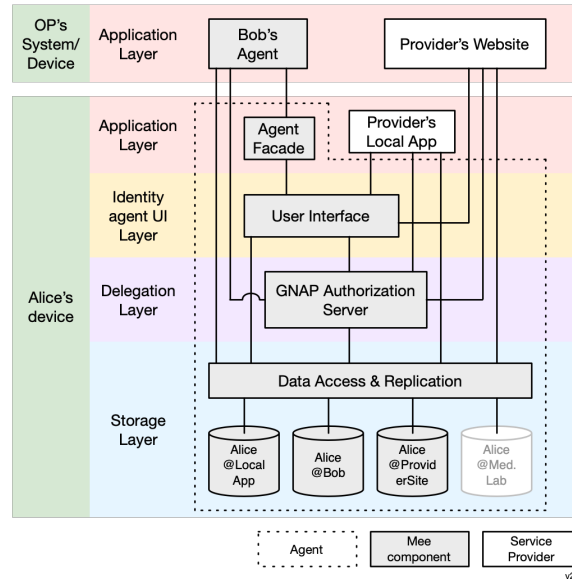
Talk here about the need for an entity to audit and enforce compliance with the legal contract]

2.12 Privacy by design

3 Architecture

3.1 Agent

The Mee architecture follows a decentralized, layered architecture shown below. We describe this architecture by considering a user, Alice, with her own device (e.g. a smart phone) as well as three other parties (OPs): a provider’s website, a provider’s local app, and another user, Bob’s, agent:



3.1.1 Application layer

Mee-compatible applications in the application layer may be service provider websites ("Provider's Website" above), other user's agents running on their own devices ("Bob's Agent"), or a service provider's local app running on Alice's device ("Provider's Local App"). Alice's agent appears to other users' agents as an app. The component supporting this on Alice's side is called the Agent Facade.

3.1.2 Identity Agent UI Layer

Alice's identity agent is deployed as an app on Alice's device. The top of layer of the agent is the UI layer that provides Alice with data management features to connect with apps/sites and manage her data. This UI allows her to inspect and in some cases edit each of the partial representations of her in each connection's context(s).

3.1.3 Delegation layer

The delegation layer handles requests for access to data from Alice's agent, local apps, remote apps, and other users' agents. In response to these requests, Alice's authorization server grants or revokes access to data in the context data storage layer.

3.1.4 Storage Layer

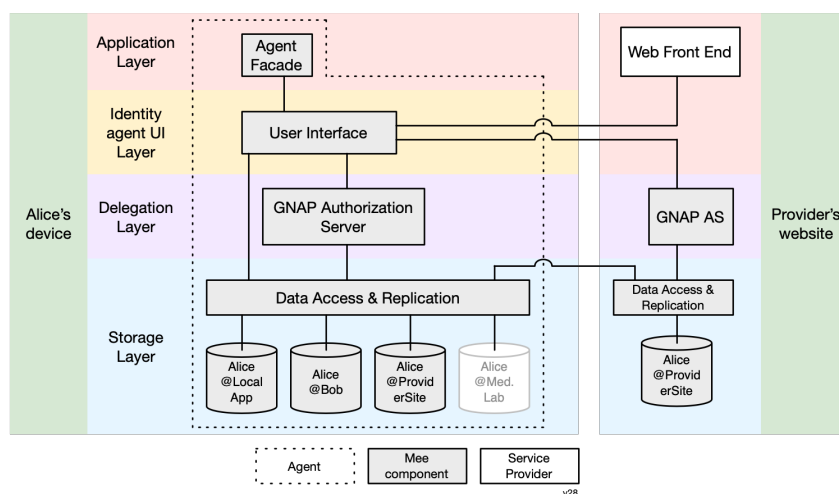
The data access and replication component provides data access (as controlled by authorization server above it) to the data in each of Alice's contexts. It manages the replication

of changes to the data in one of Alice’s contexts both (i) between the corresponding app and Alice’s agent as well as (ii) among Alice’s edge devices (phone, tablet, laptop, etc.).

This layer holds a set of contextualized representations of Alice as defined and created by apps/sites. The diagram above shows three local context data containers on Alice’s device and one, the Med Lab app’s context data container, which is not replicated on Alice’s local device (perhaps because its data set is too large for Alice’s device).

3.2 MeeTalk

Agents interact with OPs using protocols defined by open standards (e.g. OpenID Connect SIOPv2, etc.). This project is developing a protocol suite called MeeTalk that provides a technical mechanism to respect the agent user’s data rights to access, correct (editing), and delete their information held by the OP. An OP must respect these rights in order to become Mee-certified under the terms of the Mee Human Information License. One way to do this is to implement MeeTalk, although we may add support in the future for other protocols that can achieve the same result. We include a discussion of MeeTalk here on the Architecture page because it’s implementation mirrors and re-uses certain components of the agent’s architecture on the OP side. Shown below is an example of using MeeTalk to communicate between the agent and a provider’s website:



References

- [1] David Kirkpatrick. *The Facebook effect: The inside story of the company that is connecting the world*. Simon and Schuster, 2011.