Building Active Discovery

Daniel Hardman

2026-08-12

Abstract

Finding a stranger online requires an index, and an index requires that somebody catalog everyone in advance. That is why our discovery systems are surveillance systems. This document describes a way to build the alternative I have advocated for years — discovery that requires the active participation of the party being discovered — from architectures that already exist and have been implemented, measured, and in one case machine-verified. The design borrows its shape from a double-blind trial: matching is split between two parties, one holding names and attribute values but not which belongs to whom, the other holding the linkage but neither the names nor the values, so that neither can reconstruct the pairing alone, and the blind can be broken only by a defined procedure that leaves a record. Tor v3 onion services carry the conversation that follows, so neither party learns a route to the other. A verifiable attestation lets the person being sought judge whether the seeker deserves an answer, and provisional anonymity makes the seeker answerable for abuse without making them identifiable for merely asking. The document is explicit about what the design does not deliver: it does not eliminate a trusted party, it assumes honest-but-curious infrastructure, and it inherits Tor's stated limits against a global adversary.

The bind we are in

You can find almost anyone on the internet, provided somebody has already written them down. That proviso is the whole problem. An index delivers value by covering people who never asked to be covered, which means somebody collected an entry for every person who might ever be sought. The collecting is the part that privacy advocates object to. We echo the objection, and use the index anyway, because the alternative is that nobody can find anybody.

A much more restrictive posture is imaginable as an alternative: only people who have chosen to be findable can be found, and someone who has registered nothing is unreachable no matter how badly you want to reach them. For a process server or a debt collector that is a fatal limitation. For everyone whose safety depends on not being enumerable, it sounds very attractive.

I have argued in several places that this is a false choice. In a parable about Zooko's triangle, I suggested that people should discover identifiers through attributes they choose for themselves, and resolve to a secure, decentralized identifier only when a protocol actually runs [1]. In a piece about identity facets, I argued that the sameness between two facets of a life should be perceptible only to the person who owns them, revealed one relationship at a time [2]. In an analysis of correlation, I argued that the useful questions are not whether correlation happens but how expensive it is, who can do it, and how completely it links a person's separate contexts [3]. Each of those asserts a property. None of them says how to build it.

This document says how. Almost nothing in it is new, and that is the point. The matching engine was built at Microsoft Research, deployed publicly, and its privacy properties machine-checked with a theorem prover [4]. The rendezvous layer is Tor, which has carried this traffic pattern for two decades [5]. The attestation layer is whatever credential technology you already run. I contribute the assembly, the accounting of what each part does and does not give you, and one addition from an earlier paper of mine that makes the assumptions bearable [6].

Learning from blinded trials

A well-run clinical trial does not ask you to trust the people running it. It asks you to trust an arrangement. The person who assigns a patient to treatment or placebo is not the person who evaluates the outcome, and neither one can undo the blind alone. The result is credible not because the investigators are virtuous but because the arrangement narrows what any one of them can do alone.

Blinding is not secrecy. The information exists; it is partitioned so that no single party holds enough of it to act. And the blind is not permanent. A monitoring board can break it, for cause, by a defined procedure that leaves a record. That combination of partition by default and disclosure by procedure is the shape of the design that follows — what I described as reciprocal negotiated accountability in an earlier paper [6].

The analogy is not decoration. It tells you what to build and what to check. Partition the knowledge so no party can act alone. Make the unblinding procedure explicit, narrow, and auditable. Then argue about the procedure rather than about whether to have trust at all.

It also shows where this design is still weak, so I will be precise about what a partition does. It constrains: a party that holds half the picture cannot act on the whole of it. It does not by itself expose a party who cheats. A trial gets exposure from an apparatus around the blind — registered protocols, monitors, audit, the statistician who notices that the arms are unbalanced. The arrangement below has the partition and does not yet have the apparatus. That gap is the largest open item in this document, and I return to it twice.

What the design must deliver

Here are the properties an implementer should be able to check rather than just aspire to.

1. *A seeker who does not know the sought party's identifier can describe her by attributes and reach her*. This requirement rules out most of the field. Signal's contact discovery, Alpenhorn, DP5, and the rest of that family assume you already hold the other party's identifier or key [7, 8]. They are excellent systems for a different problem.

2. *No infrastructure party can determine which person holds which attributes*. Koi, the matching platform I build on below, states this as three linkages that must stay hidden, in terminology it borrows from Pfitzmann and Köhntopp [9]: that a user registered a given attribute, that some user registered two given attributes, and that two given users were matched [4].

3. *The party being sought decides, per query, whether to answer, and that decision requires nothing of her beforehand except registration*. A system that discovers you without your participation is an index with extra steps.

4. *Neither party learns a route to the other, before or after they connect*.

There is a fifth property I used to claim, and I am retracting it here, because the rest of this document does not make sense until I do. In talks and drafts I said a design like this should leave no aggregator at all: no party anywhere holding enough to answer the question, so that privacy rests on an absence rather than on anyone's restraint. I no longer think that is achievable for discovery among strangers, and the reason looks structural, not like a shortage of cleverness on my part.

Follow the argument in steps. The seeker has never met the sought party, so he holds no secret shared with her and cannot bind anything to her keys; whatever he sends is meaningful to her only after her notifiers hand it to her. Suppose she can nonetheless tell that a notification from one matcher and a notification from another belong to the same query. She must be able to, or she cannot evaluate a conjunction and the whole design fails. Whatever lets her do that is content her notifiers put there, and a notifier can always reconstruct what it sent. Nor can it be a capability special to her: no matcher knows which of its registrants are also registered elsewhere, so the recognition procedure has to work for every registrant it notifies. A procedure that works for every registrant is available to any party willing to register.

Cryptography does not rescue this, though it is fair to ask why not. Private information retrieval and private set intersection hide which record a querier touched, and they are the right tools when the querier knows what he is looking for and the parties already know each other. Neither addresses the case here, where the recognition has to happen at the recipient of an unsolicited notification, and the party constructing that notification is the party we are trying to blind.

5. The achievable invariant is weaker, and the trial metaphor already describes it. *No party can act on the linkage alone, and whoever holds a piece of it is blinded to its meaning*. That is what a split does. It is not what an absence does.

Matching that already exists

Koi is a location-privacy platform from Microsoft Research, presented at NSDI in 2012 [4]. Applications register items with attributes, and register triggers that fire when a match occurs. Attributes are general namespaced key-value pairs; location is one attribute among many, distinguished only by the platform's willingness to update it automatically.

The cloud service is split in two. The matcher knows the identities of users and the plaintext values of attributes, but not which user holds which attribute. The combiner knows the association between anonymized users and encrypted attributes, but neither the real identities nor the attribute values. A protocol between them performs matching without either learning the pairing.

Stacked negatives across two parties are hard to hold in mind. Imagine a hiring clerk with a stack of resumes from which every name has been razored out, and a receptionist with the list of who came in today and a claim ticket for each. The clerk can read every resume and say which two describe the same skill; she cannot say whose they are. The receptionist knows exactly who is in the building; she cannot read a word of what they submitted. Put them in separate rooms and they can still tell an applicant that a match came up, and neither one can tell you who is qualified for what.

Because the matcher sees plaintext attribute values, it can do semantically rich matching such as geocoding, proximity, and spelling correction — a capability that purely cryptographic ap-

proaches give up.

Those three linkages are the paper's own privacy goals, stated as things no third party may learn [4]. The second of them is the conjunctive join. It is the property I wanted and could not obtain without a fourth party, and Koi obtains it by adding one.

A couple of things about Koi are easy to miss. It is not limited to friends. The authors' social application scopes matching to friends by encrypting attribute values under a friend's key, which is an application choice rather than a platform limit; a public predicate is matchable by any stranger who constructs the corresponding trigger. And the authors checked the privacy properties with the ProVerif theorem prover [10] — modeling an adversarial matcher, an adversarial combiner, and collusion between them — rather than arguing them in prose. The public deployment performed 12,000 matches per second on a single core [4].

Koi also anticipates the objection that one combiner is a single point of trust, and answers it the way I would. Let privacy advocacy organizations, non-profits, and certificate authorities run combiners, let the user pick among many, and rely on audits, whistleblowers, and the reputational death that follows exposure. That is reciprocal negotiated accountability in embryo, nine years before I wrote the phrase.

The paper reaches for DigiNotar as its example, and the example deserves more care than either of us gave it. DigiNotar did die of exposure, so it supports the narrow claim that a trust business cannot survive being caught. It does not support the broader claim that this governance model catches anyone. The intrusion ran for weeks, the company knew and did not say, and the fraudulent certificates were finally noticed by a user in Iran whose browser had pinned Google's key — by a technical control outside the CA system, not by the audits that model relies on. Read properly, DigiNotar is an argument for the auditable notification I list among the unfinished work, and against complacency about the rest.

Since I have been calling these components off-the-shelf, I owe a caveat about availability. Koi's architecture exists and has been implemented, deployed, measured, and machine-checked; that is what this design borrows, and it is more than most proposals can point to. Koi itself is not a package you can install. Microsoft holds a patent on the architecture [11], and I am not a lawyer and offer no opinion about what it covers or who needs a license. So the honest claim is that nothing here needs inventing, not that nothing here needs building. Someone has to write the matcher and the combiner, and they should expect to read the patent first.

Rendezvous that already exists

A match is not yet a conversation, and the join between the two halves is the part an implementer will ask about first. It works because of a property of onion addresses that is easy to overlook: an onion address is a rendezvous capability, not a route. It names a service without locating it, and no amount of study tells you where the machine is.

That makes the handoff one-directional and cheap. The seeker stands up a fresh onion service for this search and puts its address in the trigger he registers. When a match fires, the notification the sought party receives carries that address. If she decides to answer, she dials it from her own client, over her own circuits; if she wants a return path, she stands up an onion service of her own and offers it inside the conversation, after she has decided the seeker is worth talking to. Neither party ever hands over anything that resolves to a location, so requirement 4 survives even though

the seeker's address passes through the matcher in the clear. The infrastructure must never learn which registrant dialed, and that is a property of the matching split rather than of the transport.

Two consequences are worth stating. The seeker's address is public to anyone who sees the query, so it should be disposable and abandoned when the search ends. And anyone holding it can test whether the service is up, which leaks the seeker's presence but not the sought party's.

Once two parties have agreed to talk, they need a channel neither can trace to the other. Tor v3 onion services provide it.

A service builds circuits outward to introduction points and holds them open. A client sends an introduction request to one of those points, which relays it down the circuit the service established. Neither side learns the other's address. The service's descriptor is stored at directory nodes chosen by a blinded public key derived from the service's identity key and the current time period, so a directory node holds a rotating pseudonym rather than an address. The descriptor is encrypted such that a reader must already know the onion address to decrypt it [5].

That design was adopted deliberately to fix a harvesting attack. Under version 2, directory nodes learned the plaintext address of every service that published to them, and adversaries ran directories at chosen positions to collect addresses and probe them. The Tor Project tracked this as a defect and closed it on the strength of key blinding [12]. An earlier paper of mine proposed splitting an identifier into fragments and registering each with a different rendezvous node, which invites exactly this attack. Tor's answer is better than mine was, and this document is where I stop trying to improve on it.

[SUGGESTED DIAGRAM: Rendezvous without routes — two parties, each holding an outbound circuit to its own introduction point, meeting at a rendezvous point neither controls, with the directory node holding only a blinded key and no address.]

Proving who is asking

The person being sought needs to judge whether the seeker deserves an answer. A stranger claiming to be a journalist is making a claim, and a claim from a stranger is worth what you paid for it. So the seeker attaches evidence of who he is, and the sought party evaluates it before deciding.

The requirement matters more than the technology, and it has three parts. The evidence must be verifiable by a stranger without a callback to its issuer, because a callback tells the issuer that a verification is happening and when. It must remain checkable later, since the decision may be litigated long after the keys involved have rotated. And it must support chaining, because real authority is delegated in steps: an editor authorizes a reporter, an institution authorizes the editor, and increasingly some of those steps are taken by software acting for a person.

I recommend KERI with ACDCs [13, 14], for reasons I have developed elsewhere rather than a preference for the local tribe.

Evidence outlives keys. A certificate is a dynamic privilege mechanism whose lifespan is shrinking toward weeks, while the relationships it attests are stable and measured in decades; renewing an identity is a non sequitur [15]. Certificate Transparency improved the accountability of that model without changing where its trust bottoms out [16]. A verifier usually needs historical provability rather than present-tense key validity: what was valid then, not only what is valid

now [17]. Binding evidence to a rotating identifier rather than to a key makes that answerable. It is the difference between an attestation you can still evaluate in five years and one you cannot.

Chaining survives rotation. Where authority is bound to a key, rotating the key breaks the delegation, and every downstream grant has to be reissued. KERI-style delegation binds to the identifier, so rotation does not disturb the chain. That matters more as chains get deeper, and it keeps multi-step delegation to software agents off the reissuance treadmill. The full structure such a chain needs — which acts, on whose behalf, in whose interest, under whose liability, revocable how — is the subject of separate work of mine. The part that bears on discovery is that verification is open-loop: a stranger can check authority without consulting its issuer [18].

The post-quantum path is already designed. KERI's pre-rotation commits to the next key with a quantum-safe hash rather than exposing it, so control survives even an adversary who later breaks the signing algorithm. Its cryptographic agility allows migration without a global flag day [19]. For evidence meant to be evaluated in a decade, that is not a theoretical nicety.

The alternatives all work, and I list them in the order I would reach for them. OpenID for Verifiable Presentations defines a mechanism for the verifier to authenticate itself to the wallet before the holder releases anything, including an attestation issued by a party the wallet trusts [20]. It is finalized, widely implemented, and the most likely thing to already be running in an enterprise deployment. W3C Verifiable Credentials give the broadest tooling ecosystem, and carry properties I have criticized at length elsewhere [21]. DIDComm gives asynchronous, transport-agnostic messaging secured by DID control, which fits the rendezvous pattern well [22]; I have written about how it differs from the others [23].

All of them satisfy the part of the requirement that the architecture depends on: a stranger can check the evidence without a callback. They differ on the two parts I argued for above, since evidence bound to a key gets harder to evaluate once the key has rotated, and a delegation chain built that way has to be reissued when any link rotates. Those costs fall on the deployment rather than on the design, so a shop that already runs one of these should use it and know what it is paying.

Two cautions follow. The evidence the seeker presents is identical at every party that sees it, which makes a conventional signed presentation a durable correlator across matchers and across queries. If the seeker shows the attestation to more than one infrastructure party, he needs to present it unlinkably, and unlinkable presentation that still permits counting is not a solved problem. The narrower and safer arrangement is for the seeker's evidence to travel end-to-end and be evaluated only by the sought party. Separately, an alias for an identifier is not a proof about the party behind it, and the most dangerous mistake available here is confusing what you have proved with what you have merely guessed [24].

Accountability that runs both ways

Requiring the seeker to identify himself protects the sought party and endangers the seeker. A journalist in a hostile jurisdiction who must announce herself to infrastructure to look for a source has swapped the risk she was avoiding for a new one. And an attacker who can afford credentials is not deterred by needing them.

I considered pricing queries, so that discovery would cost the seeker something the subject receives, and I have come around to thinking it does not carry the weight I wanted from it. The

obvious objection to my objection deserves stating first, because it is strong. Per-query tolls do screen abusers, and that is the whole logic of hashcash and of Tor's own client puzzle, which advertises a difficulty in the descriptor and queues introductions by the effort a client spent [25]. A spammer's value per message is a fraction of a cent, so a toll of a fraction of a cent ruins him and costs an honest sender nothing.

The trouble is that a discovery enumerator is not a spammer, and the two differ on both terms. His value per hit is not negligible: located-person records trade in the data-broker market at cents to tens of cents apiece, and a toll set below that is an operating expense rather than a deterrent. Meanwhile the scarce resource here is not server capacity, which scales with money, but the attention of the person being notified, which does not scale at all. A predicate can support only so many notifications a day before its registrants stop reading them, and when supply is fixed a clearing price does not suppress demand. It allocates the supply to whoever bids highest, and that is the adversary. Tor's puzzle is a flood defense, shipped disabled by default, that advantages whoever owns the most compute.

Provisional anonymity sorts the honest seeker from the enumerator [6]. A seeker walks in as a stranger and stays one unless he misbehaves, at which point he was never as anonymous as he looked.

The mechanism has three moves. The seeker encrypts his identifying information, using verifiable encryption — a scheme that lets him prove things about the plaintext without revealing it, so the infrastructure can confirm he sealed up a real identity rather than a string of zeros, without learning whose. He lodges the decryption key with an independent escrow. Then he queries anonymously. If he abuses the privilege by enumerating the population or harassing the people he finds, the escrow releases the key on a showing of cause and he is named. Neither the matching service nor the escrow can unmask him alone, and the escrow cannot tell what any key it holds is for, so it can be audited aggressively without endangering anyone.

Accountability after the fact, rather than a toll in advance, gets the incentives right. It does not advantage the wealthy. It creates no payment channel to correlate. The journalist who merely looks stays anonymous; the one who abuses the system does not. And the conditions under which unmasking may occur can travel inside the query, bound to it so they cannot be stripped. I have elsewhere called that a watermark [6]: not concealed information in the steganographic sense, but terms that any later use of the seeker's identity has to carry along.

The word "alone" in "neither can unmask him alone" is carrying more weight than one word should, and the paper that introduced this technique did not say enough about it either. Two parties who cannot act alone are still one court order apart if they answer to the same authority. So the escrow's independence is jurisdictional before it is technical: an escrow incorporated where the matcher is incorporated buys almost nothing, and the arrangement is worth what its least independent party is worth. Splitting the escrowed key across several holders in several jurisdictions raises the cost of coordinated compulsion, which is the sharding my earlier paper suggests and does not develop [6]. Even so, a seeker whose adversary is a state that can reach every party in the arrangement should not rely on this at all. That seeker wants the broadcast arrangement in Appendix A, which has nobody to compel.

This is the monitoring board breaking the blind. It is the same move, with the same justification, and it is why a partition is more useful than an absence.

What this does not give you

The design assumes honest-but-curious infrastructure. Koi's threat model requires that the external interface of each party be conformant, and its authors are explicit that the assumption rests on a commercial reality — a large provider fears the backlash from externally visible misbehavior — rather than on a cryptographic guarantee [4]. A malicious matcher that violates conformance can do real damage. If it chooses which registrants to notify rather than notifying all of them, it can test a hypothesis about a single person: notify only the target on one leg of a two-attribute query, and an answer proves the target holds the other attribute. Koi anticipates a cousin of this and answers with detectability, observing that a matcher registering fake users risks exposure when a real user notices being matched with one [4]. That instinct is right, and it is not a proof. Closing the gap properly requires notification that can be audited, so a registrant can verify she was not silently excluded, and that is genuine work rather than configuration.

Individually sensitive attributes are out of scope. Koi assumes the attributes themselves are not sensitive, and that only the linkage between a user and an attribute needs protecting [4]. That assumption holds for a taste in food and fails for a medical diagnosis. A matcher holding a single sensitive predicate is a target list whatever the pseudonyms around it, and a general deployment needs a separate answer for those, which this document does not have.

Tor does not defend against a global passive adversary, and says so: like all practical low-latency systems, it does not protect against an adversary who can observe the whole network [26]. A later analysis shows a residual leak in the lookup itself: a client's request to a directory carries the blinded key for the current period in the clear, so a directory that already knows a target address can detect and correlate lookups for it [27]. Systems that do defend against a global passive adversary exist, and their price is instructive. Alpenhorn reports 10 million users at about 3 KB per second of standing traffic for dialing, which its authors work out to 7.8 GB per user per month, with dial latency around 150 seconds [7]. That cost is not an implementation shortcoming. Against such an adversary, latency overhead and bandwidth overhead trade against each other under a proven lower bound, so buying strong anonymity cheaply in both currencies at once is not available [28]. None of these systems runs as a service anyone can join.

Retrieval leaks even when storage does not. Apple's offline finding network is the largest deployment of the pattern where a subject fetches something indexed by a key only she can derive. An independent analysis found that the design achieves finder anonymity and report confidentiality, yet owner devices authenticate to fetch, which lets Apple correlate retrievals to owners [29]. Whatever mechanism notifies a sought party here deserves the same scrutiny, because the fetch is as revealing as the record.

Non-collusion remains an assumption. It is instrumented rather than eliminated, through many combiners, user choice among them, audit, and the reputational consequence of exposure. That is a governance claim, and it deserves to be argued as one rather than dressed up as a cryptographic one.

Discovery is not fast. The sought party answers when she chooses to. The channel can be quick, but her decision is human, and it will take minutes to days. This is a property to own rather than apologize for, since the deliberateness is where the consent comes from.

The design also keeps no history. You cannot ask what was true last year, which is a genuine loss for research, audit, and journalism, and an unavoidable consequence of storing nothing that

could answer.

What remains to be built

Five pieces are missing.

1. Auditable notification, so that selective notification by a malicious matcher becomes detectable rather than excluded by assumption. This is the most important open item, and I should be straight about its status: it is a research problem, not a backlog ticket. Making a party prove it told everyone, to recipients who must stay unlinkable to it and to each other, is not something I can point you at a solution for. Transparency logs and verifiable broadcast are the neighborhoods to search in. Until someone solves it, the design rests on the honest-but-curious assumption rather than on structure, which is the gap I flagged when I set up the analogy.
2. A treatment for sensitive predicates, whether by thresholds, decoys, or refusing to index them at all.
3. An unlinkable presentation of the seeker's attestation that still supports counting, if the attestation must be seen by infrastructure rather than only by the sought party. Anonymous rate-limiting tokens are deployed for a related problem [30] and do not by themselves solve this one.
4. A schema for describing what a seeker wants. This is the one place where prior art of my own already exists. An Aries protocol I co-authored in 2018 specifies how one party asks another for help discovering an unknown person, with nested boolean criteria and per-criterion identifiers so a response can report which criteria matched [31], and a companion protocol supplies the double opt-in [32]. Neither notifies the sought party nor seeks her consent, since the responder exercises judgment on her behalf, and that is exactly the gap this design fills. The criteria language is reusable.
5. Governance for the combiner population: who may run one, what an audit consists of, and what happens when collusion is found.

Only the first of those is research. The rest is engineering, and the parts it connects have all been built before, which is a better position than this problem has been in for the twenty years I have watched people try to solve it.

Here is what changes if someone does the work. Today a person who wants to be reachable has to be listed, and being listed means being enumerable by anyone who buys the list. Those have always arrived together, and we have argued about how much surveillance to tolerate as the price of being findable at all. They do not have to arrive together. A person could publish an attribute, wait, and answer the one stranger in a year who has a reason she accepts — and no party anywhere, including the ones running the service, would be holding a list of people like her. That is not a small change in the economics of surveillance. It removes the corpus that surveillance presupposes, which is why I keep coming back to this problem.

Appendix A: two fallbacks

The design above assumes a population large enough that scanning it is impractical, and a threat model in which honest-but-curious infrastructure is acceptable. Two simpler arrangements apply

when those assumptions fail.

Broadcast removes the infrastructure entirely. The seeker posts an encrypted call to an append-only board, and every participant fetches everything and decrypts locally. There is no query to correlate, no matcher to collude, and no linkage of any kind, because nothing is routed per subject. The cost is that every participant downloads the whole corpus, which is prohibitive at consumer scale and free at small scale. For a few thousand calls a year among people facing a serious adversary, this is the right answer and the simplest to build. Ateniese and colleagues implemented matchmaking encryption in this shape over Tor onion services, with both parties specifying policies for each other and nothing leaking but the fact of a match [33]. Its weakness is that the authority can decrypt everything, so it suits a deployment where the sensitive direction carries no credential.

Bucketing sits between the two. Participants fetch a bucket rather than the whole corpus or a single record, with the bucket determined by a prefix or hash of the attribute. The leak is bounded by the bucket population, and it is a parameter you state rather than a property you hope for. Tor's own directory works this way, and the widely deployed instance of the pattern is the range query used to check a password against a breach corpus, where a client sends a hash prefix and receives every matching suffix [34]. Bucketing is the pragmatic middle: cheaper than broadcast, weaker than broadcast, and honest about the difference as long as the parameter is published.

Choosing among the three is a threat-model question. Broadcast when the population is small and the adversary is strong. The split-matcher design when the population is large and the infrastructure can be held accountable. Bucketing when neither extreme fits and you are willing to publish the bucket size.

References

[1] Hardman, D. Zooko and Houdini: a Flatland parable. 2024. https://dhh1128.github.io/papers/zh.html

[2] Hardman, D. Identity facets. 2026. https://dhh1128.github.io/papers/if.html

[3] Hardman, D. The dangerous half-truth of "we'll be correlated anyway". 2020. https://dhh1128.github.io/papers/wbca.html

[4] Guha, S., Jain, M., and Padmanabhan, V. N. Koi: A location-privacy platform for smartphone apps. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '12*), San Jose, CA, April 2012. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nsdi12-koi.pdf

[5] The Tor Project. Tor rendezvous specification, version 3. https://spec.torproject.org/rend-spec/

[6] Hardman, D. A call for reciprocal negotiated accountability. 2021. https://dhh1128.github.io/papers/crna.html

[7] Lazar, D. and Zeldovich, N. Alpenhorn: bootstrapping secure communication without leaking metadata. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI '16*), 2016. https://pdos.csail.mit.edu/papers/alpenhorn:osdi16.pdf

[8] Borisov, N., Danezis, G., and Goldberg, I. DP5: a private presence service. *Proceedings on Privacy Enhancing Technologies* 2015, 2 (2015), 4–24. https://petsymposium.org/popets/2015/popets-2015-0008.pdf

[9] Pfitzmann, A. and Köhntopp, M. Anonymity, unobservability, and pseudonymity — a proposal for terminology. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 1–9, 2001.

[10] Blanchet, B. ProVerif: automatic cryptographic protocol verifier. https://bblanche.gitlabpages.inria.fr/proverif/

[11] Guha, S., Padmanabhan, V. N., Jain, M., and Jain, A. Privacy-preserving matching service. US Patent 8,868,654 B2, Microsoft Technology Licensing LLC, filed June 6, 2011, granted October 21, 2014. https://patents.google.com/patent/US8868654B2/en

[12] The Tor Project. Make .onion addresses harder to harvest by directory servers. Issue 8106. https://gitlab.torproject.org/legacy/trac/-/work_items/8106

[13] Trust Over IP Foundation. Key Event Receipt Infrastructure (KERI) specification. https://trustoverip.github.io/kswg-keri-specification/

[14] Trust Over IP Foundation. Authentic Chained Data Containers (ACDC) specification. https://trustoverip.github.io/kswg-acdc-specification/

[15] Hardman, D. Why X509 certs should be secondary evidence of org identity. 2024. https://dhh1128.github.io/papers/x509-prob.html

[16] Hardman, D. Where trust bottoms out: X.509, Certificate Transparency, and KERI's DPKI architecture. 2026. https://dhh1128.github.io/papers/wtbo.html

[17] Hardman, D. What does telco need? Requirements for organizational identity evidence. 2026. https://dhh1128.github.io/papers/telco-ev-reqs.html

[18] Hardman, D. The shape of delegated authority. 2026. https://dhh1128.github.io/papers/sda.html

[19] Hardman, D. KERI's strategy for post-quantum security. 2025. https://dhh1128.github.io/papers/kspqs.html

[20] OpenID Foundation. OpenID for Verifiable Presentations 1.0. Final, July 2025. https://openid.net/specs/openid-4-verifiable-presentations-1_0.html

[21] World Wide Web Consortium. Verifiable Credentials Data Model v2.0. https://www.w3.org/TR/vc-data-model-2.0/

[22] Decentralized Identity Foundation. DIDComm Messaging v2. https://identity.foundation/didcomm-messaging/spec/

[23] Hardman, D. Sentries, confessionals, vaults, and envelopes. 2023. https://dhh1128.github.io/papers/svce.html

[24] Hardman, D. Opaque identifier aliases. 2026. https://dhh1128.github.io/papers/oia.html

[25] The Tor Project. Proposal 327: a first take at PoW over introduction circuits. https://spec.torproject.org/proposals/327-pow-over-intro.html

[26] Dingledine, R., Mathewson, N., and Syverson, P. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004. https://www.usenix.org/legacy/event/sec04/tech/full_papers/dingledine/dingledine.pdf

[27] Improving the privacy of Tor onion services. IACR Cryptology ePrint Archive, Report 2022/407, 2022. https://eprint.iacr.org/2022/407

[28] Das, D., Meiser, S., Mohammadi, E., and Kate, A. Anonymity trilemma: strong anonymity, low bandwidth overhead, low latency — choose two. In *IEEE Symposium on Security and Privacy*, 2018. https://eprint.iacr.org/2017/954

[29] Heinrich, A., Stute, M., Kornhuber, T., and Hollick, M. Who can find my devices? Security and privacy of Apple's crowd-sourced Bluetooth location tracking system. *Proceedings on Privacy Enhancing Technologies* 2021, 3 (2021), 227–245. https://petsymposium.org/popets/2021/popets-2021-0045.pdf

[30] Internet Engineering Task Force. The Privacy Pass Architecture. RFC 9576, June 2024. https://www.rfc-editor.org/rfc/rfc9576.html

[31] Aristy, G. and Hardman, D. Aries RFC 0214: "Help Me Discover" protocol. Start date August 20, 2018. https://github.com/decentralized-identity/aries-rfcs/blob/main/features/0214-help-me-discover/README.md

[32] Hardman, D., Curren, S., Curran, S., Looker, T., and Aristy, G. Aries RFC 0028: Introduce protocol 1.0. 2019. https://github.com/decentralized-identity/aries-rfcs/blob/main/features/0028-introduce/README.md

[33] Ateniese, G., Francati, D., Nuñez, D., and Venturi, D. Match me if you can: matchmaking encryption and its applications. In *Advances in Cryptology — CRYPTO 2019*. IACR Cryptology ePrint Archive, Report 2018/1094. https://eprint.iacr.org/2018/1094

[34] Have I Been Pwned. Pwned passwords range query API. https://haveibeenpwned.com/API/v3#PwnedPasswords