Open Network Handles

—

Design Notes
(DRAFT 0.3)

Michael J. O'Donnell

17 August, 2002

Abstract

I propose a system of *Open Network Handles* to provide permanent primitive network handles promiscuously to all who request them. Handles provide an intermediate level of service between IP numbers and domain names. While assignment of IP numbers is constrained by routing considerations, the owner of a handle may reassign it to different addresses over time for mobility or changes p in configuration of resources. Unlike domain names, handles carry no significance in natural language, so they should not have high commercial value, nor should they attract disputes based on assertions of rights in significant names.

This document describes several considerations and questions that arise when planning a design of a network handle system.

0.1 For now, I'm just listing issues as they occur to me.

0.2 Still a chaotic list, I just added and refined a bit, and converted from "central" to "global."

0.3 Minor refinements.

1 Functional design

I have no background in cryptographic key management. I hope that someone with such knowledge will contribute.

- Although the focus of *service* is to answer queries by resolving handles to addresses, the focus of *design* is the establishment of ownership. Assuming that ownership is established by knowledge of a secret key, the focus of design is resolving each update request into a challenge to the appropriate secret key. With public-key techniques, the focus of design is the association of handles with public keys.

- A good design should add very little overhead to satisfactory communications between queriers and handle owners who decide to trust one another. The network handle system should avoid interfering with such communications. It should aid queriers and handle owners to establish communications, and it should provide methods for authenticating contacts when they are challenged.

- A good design should separate *discovery* and *authentication* as much as possible.

- Should handles be related systematically to authentication keys, or should they be assigned independently? If handles are determined by keys, we avoid attempts to catch numerologically significant handles, and allow more end-to-end verification. But transfer and upgrade are less efficient and more vulnerable when we can't change the key for an existing handle. On the other hand, the benefits for transfer and upgrade may require a central authority, and we might be able to avoid such an authority entirely with handles generated from keys. And accommodation of naive owners seems to require that we allow weak password protection, which doesn't appear to support an enforced key/handle correspondence. The global system could keep cryptographic keys on behalf of naive users, and provide weak password access.

- Perhaps we can allow arbitrary handle/key pairs, while still providing permanent certificates of authenticity. If two agents claim the same handle, they can produce time-stamped certificates from a trusted certifier. The trusted certifier doesn't need to survive (in fact demise and destruction of the private keys is an advantage to confidence in authenticity). But a querier must locate the proper owner of a handle in order to detect a fraudulent one. What if each querier keeps the handle plus its time stamp? We might also depend on a certifier who certifies uniqueness, and not just timeliness. But that sounds like a substantial burden on the continuity of the certifier. Does it support fully transferable handles?

- The current DNS is a very successful and helpful proof of concept for most of the design problems having to do with storing tables and resolving queries.

- The system should support the most flexible possible notions of virtual agents. In particular, an individual application running on a host should be an agent, as well as the host in its entirety. The system should also support distributed agents, mobile agents, etc. Support for virtual agents appears mainly to call for generalized addresses.

- In addition to normal addresses (which, at least to start, can be exactly the sorts of addresses allowed by DNS), perhaps handles should be able to resolve to other handles, constituting *permanent reassignments*. As much as possible, a permanent reassignment should be irrevocable. This

provides something near the utility of a transfer of handle. It can also be the basis for upgrade of authentication/cryptographic techniques. It works even if handles are rigidly determined from keys.

- In deciding whether to implement a particular extension of the scope of addresses, the main consideration should be whether the extension can be supported well by direct communication between querier and handle owner. For example it makes a lot of sense to support hunt groups of IP addresses, since a querier needs the information in a hunt group precisely when he cannot reach the handle owner at a single IP address. But it makes less sense to provide network-extraneous information about the handle owner, since the owner may send such information directly to the querier.

- Transfer by reassignment of an independently assigned handle is enforceable in the contractual sense. The receiver of such a transfer may retain a signed certificate of transfer, may send in test queries from time to time, and may seek relief for misdirections by the source of the transfer (perhaps this requires *signed* responses to queries). But it will be nicer to have positive enforcement, in the sense of making it technically difficult for the source of a transfer to renege. That sounds difficult with independently assigned handles. Transfers might also be achieved through an escrow service, but that adds cost and also adds another agent who must be trusted.

- In terms of logical functionality, handles are pure atoms with no meaningful relations to one another besides equality and inequality. But the administration of handles imposes other relations, such as a hierarchy of handle authorities, each represented by a handle. The reassignment relation above is another sort of administrative relation between handles. We should minimize the number and complexity of relations that need to be supported by the handle system itself.

- A hierarchical system is very attractive. In principle, it can be achieved merely by implementing a top-level system of unique handles, which may themselves refer to local handle resolvers. The independence of local regions within handle space, allowing extra services to be provided for local needs, is attractive. But there should be a protocol which, if followed by the agents involved, treats lower-level handles uniformly and transparently with top-level handles for the purposes of query resolution.

- The natural structure of networks, and particularly of the Internet, makes it relatively feasible and efficient to guarantee that correct responses to queries will eventually come, but relatively difficult to guard against incorrect responses.

2 Incentives

Compare the value of a globally assigned handle to the value of an independently assigned handle:

Property	**Global**	vs.	**Independent**
Control	limited	$<$	total
Key management	multiple keys	$<$	one key
Size of handle	moderate	$\approx$	a few bits longer
Support	from registrar	$>$	from self
Transferability	total	$>$	requires additional escrow

Table 1: Comparative value of globally assigned vs. independently assigned handle

- The main incentives for registering multiple handles globally instead of independently appear to be

  1. Let the global system bear the burden of running the query server.
  2. Maintain independent transferability.

  On the other hand, flexibility and control actually favor independent assignment, as well as key-management load. But the latter incentive has little impact on naive users who opt for password protection, and those are the most likely users to want to unload query resolution burden.

- To reduce the incentive to let the global system provide multiple handles, avoiding the need for local support, we need effective distribution of handle system software.

- We can reduce or eliminate the incentive to grab many handles from the global system so that they are independently transferable, if we let agents promote their independently assigned handles to the global system for the purpose of transfer. This complicates the top-level tables, and makes it more difficult to enforce a specific correspondence between handle and key. We might allow promotion for the *sole* purpose of escrowed reassignment with a timeout.

3 Vulnerabilities

3.1 Exposure to Harm

- Because the handle system makes no attempt to authenticate the quality of handle owners, all harm due to bad behavior by a handle owner must be avoided outside of the handle system.

- Denial of handle traffic to the owner-authorized address. This sort of harm probably cannot be defended by any sort of end-to-end verification.

- Duplication of handle traffic to a fraudulent address. If we allow multiple-delivery addresses, or if the attacker diverts but also forwards, this may occur without denying traffic to the authorized address. For the purposes of spying, duplication may be more harmful than denial. Querier and owner may defend by end-to-end encryption, but the spy may still derive some benefit from partial information.

- Diversion of handle traffic to a fraudulent address. This is just the combination of the previous two. It may be harmful even if the target of the diversion belongs to the handle owner. Diversion is only a problem when a proper handle owner is disenfranchised after confirmation of a handle assignment. Hijacking of an initial handle assignment is no harm (except possibly a denial of new-handle service if it is repeated consistently), since it doesn't matter who gets which handle.

- Creation of traffic to an address whose owner doesn't want it. There are so many other ways to accomplish this, that the additional possibility of doing it through the handle system is probably not very important.

- Denial of a new handle to an agent who requests one. This is only a temporary harm, but if repeated consistently it can destroy the utility of the system.

3.2 Attacks and Defenses

I have no background in threat analysis. I hope that someone who understands such things will contribute.

- Attack by cracking the cryptographic authentication system and entering a fraudulent address. This is detectable by the owner. Partial defense: allow the owner to invalidate a compromised handle. The attacker can also invalidate the handle, but this reduces the harm from redirection to denial. Partial defense: provide audit trails that are hard for an attacker to cover, even when he has compromised the handle. E.g., notify a certain address of every update to a given handle.

- Attack by cracking the authentication system and changing the key. This is deadly if we allow liberal changes of key. The possibility of this attack supports the decision to require reassignment, which is highly auditable by lots of parties, as the only way to change a key.

- Attack by discovering a private key through channels having nothing to do with cryptography. Key management was, is, and will probably remain the weak spot in cryptographically enforced security. The defense appears to be the same as the defense against key cracking.

- Accidental self-attack by losing a private key. Any attempt to defend against this sort of self-attack appears to weaken the defense against malicious outside attack. It is probably best to leave this sort of defense completely out of band. If the problem is large enough, a trusted agent might charge a fee to register replacements for lost handles, based on out-of-band verification of identity.

- Accidental self-attack by announcing a reassignment, or other irrevocable change. Defend by requiring confirmation before committing irrevocable change.

- Attack by tricking a key owner into muffing key management or making an irrevocable change. This is similar to the mental viruses that induced computer owners to delete files by fraudulently associating those files with computer viruses. The defense appears to be good training, which depends critically on keeping things simple, but exposing users to key-management details in gentle doses. Seems pretty challenging.

- Attack by intercepting and modifying communications with servers. I hope that this has been studied from the point of view of other services. It doesn't appear to constitute a special threat to a handle system. Defense: end-to-end authentication.

- Attack by flooding query servers with queries. This appears to be the same as any denial-of-service by flooding attack.

- Attack by flooding a registrar with requests for new handles and/or updates. Also appears to be the same as any denial-of-service by flooding attack.